Integrating Shape-from-Shading & Stereopsis

Tom S. F. Haines

Submitted for the degree of Doctor of Philosophy

Department of Computer Science

THE UNIVERSITY of York

2009

Abstract

THIS thesis is concerned with inferring scene shape by combining two specific techniques: *shape-from-shading* and *stereopsis*. *Shape-from-shading* calculates shape using the lighting equation, which takes surface orientation and lighting information to irradiance. As irradiance and lighting information are provided this is the problem of inverting a many to one function to get surface orientation. Surface orientation may be integrated to get depth. *Stereopsis* matches pixels between two images taken from different locations of the same scene - this is the correspondence problem. Depth can then be calculated using camera calibration information, via triangulation. These methods both fail for certain inputs; the advantage of combining them is that where one fails the other may continue to work. Notably, shape-from-shading requires a smoothly shaded surface, without texture, whilst stereopsis requires texture - each works where the other does not.

The first work of this thesis tackles the problem directly. A novel modular solution is proposed to combine both methods; combining is itself done using Gaussian belief propagation. This modular approach highlights missing and weak modules; the rest of the thesis is then concerned with providing a new module and an improved module. The improved module is given in the second research chapter and consists of a new shape-from-shading algorithm. It again uses belief propagation, but this time with directional statistics to represent surface orientation. Message passing is performed using a novel method; it is analytical, which makes this algorithm particularly fast. In the final research chapter a new module is provided, to estimate the light source direction. Without such a module the user of the system has to provide it; this is tedious and error prone, and impedes automation. It is a probabilistic method that uniquely estimates the light source direction using a stereo pair as input.

Acknowledgements

I will start by thanking chaos, for accidentally letting life form on this planet, and then not killing it off with a large rock. Further to this, specific consideration should be given to the entire chain of organisms leading up to, and including, my parents for breeding in the specific ways in which they did. A special mention has to be made for the creatures in the general vicinity of these many organisms that failed to eat anything on the aforementioned parent chain before it reproduced. Their general uselessness has been wholly to my advantage.

Much greater importance has to be given to all the animals that have collected and shared knowledge over the centuries, constructing the intellectual foundation for me to add my speck of dust to. A specific mention has to be made of Richard, my supervisor, for helping me absorb enough knowledge to construct something new, if only in this small corner of the universe. This thanks is extended in some measure to the many people I have had interesting conversations with, including both assessors, other PhD students and many, many more. Particular mention in this regard goes to the York university philosophy society for keeping me sane, or at least providing an insane environment where, relatively speaking, I feel as such.

Friends and family, as scattered and far away as they sometimes are, have contributed greatly to the state of my existence, my happiness and my chances to get inebriated at a good party. In case I have missed anybody I would like to thank the entirety of history, the universe and the laws of physics, though I am not entirely sure what for specifically. Whilst they are the source of all problems it is solving these problems that makes existence anything but dull.

Contents

List of Figures

Declaration

I declare that the work in this thesis is solely my own except where attributed and cited to another author, with the exception of this declaration. Most of the material in this thesis has been previously published by the author. A complete list of publications can be found over the page.

Publications

- T. S. F. Haines and R. C. Wilson, *Integrating Stereo with Shape-from-Shading derived Orientation Information*, British Machine Vision Conference, 2007.
- T. S. F. Haines and R. C. Wilson, *Belief Propagation with Directional Statistics for solving the Shape-from-Shading problem*, European Conference on Computer Vision, 2008, pp. 780–791. (Oral)
- T. S. F. Haines and R. C. Wilson, *Combining Shape-From-Shading and Stereo Using Gaussian-Markov Random Fields*, International Conference on Pattern Recognition, 2008, pp. 1–4.

Chapter 1
Introduction

CORE to this work are three related algorithmic ideas, two of which are the subject of the title. The third, light source estimation, is essential to constructing a complete system. Before continuing definitions are given:

- *Shape-from-Shading* [SfS]. Input is one image of a scene and a function from surface orientation to irradiance, often referred to as the *reflectance map*. Output is scene shape. Irradiance is provided by the image, so the reflectance map needs to be inverted to get surface orientation. Ambiguity exists however as the reflectance map is many to one. Surface orientation may be integrated to get depth, either separately or as part of the algorithm.

- *Stereopsis*. Input is two images of a scene; output is again scene shape. Key is solving the *correspondence problem* - that of matching pixels projected onto each image from the same scene point. Due to two view geometry the matches have to be on lines, making this a $1D$ search problem. *Rectification* can transform the images so the search is along scan-lines. Two view geometry also provides triangulation, which converts matches into depth.

- *Light Source Estimation* [LSE]. Unlike the previous this is better described as a class of algorithms, with many possible inputs/outputs. Input is often both an image and the shape of the scene. Output is some description of lighting - anything from a single directional light to multiple area lights. As the SfS reflectance map is itself a function of the light source a certain symmetry exists with SfS. In both you invert a known process with unknown inputs, but SfS uses the light source as an input and surface orientation as an output, whilst LSE does the reverse, swaping the inputs and the outputs.

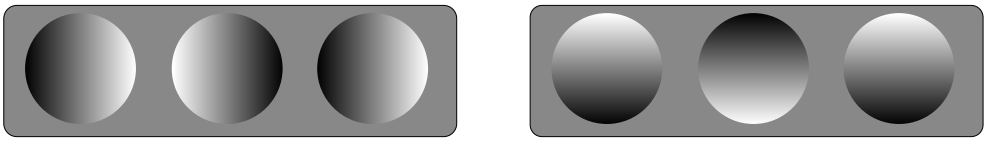


Figure 1.1: Images which the human brain can interpret as being in two different states. The left triplet is easy to consciously switch between the concave/convex states, whilst the right triplet is harder due to the human preference for light emitting from the sky, so the two outer circles prefer to remain convex.

The goal of this thesis is to combine the first two approaches above - SfS and stereopsis; in consequence it is highly desirable to also include the third approach, light source estimation. Justification follows from examining the interplay of the first two approaches. SfS requires knowledge of the equation mapping surface orientation to irradiance. This equation can normally be broken down into three parts - light source information, a shading model and shading model parameters. The shading model parameters, most typically albedo, have to be provided to the algorithm, and are most commonly set constant for the entire scene, though most algorithms will accept parameters per pixel if you can infer them. Correspondence, the basis of stereopsis, involves matching patterns - if there is no pattern to match then stereopsis fails, usually by fitting a plane or a fronto-parallel plane[1]. The issue is that constant shading parameters do not create patterns that stereopsis can match, and so if we want stereo to provide much useful information we need an image that contains areas with variable albedo. It is unreasonable to consider an albedo map as a simple input, as its inference is an involved process. Light source estimation algorithms that do not take albedo as an input will commonly provide it as an output, and using one has the added convenience of also providing light source information. Alternatively albedo can be separately estimated once the light source direction is known. Given this the input to all three techniques combined is simply two calibrated images, as the shading model is usually fixed in the algorithm's design. This is far more reasonable than expecting light source and albedo information to be provided.

Given the complexities of combining algorithms it is reasonable to consider the value of combining SfS and stereopsis, for which two arguments are offered.

[1] A plane which is parallel to the viewing plane, i.e. of constant depth from the viewer in an orthographic projection.

Firstly, there is an evolutionary argument. Many animals have two eyes, and whilst some use their eye pairs for 360° vision many, especially predators, use stereopsis to obtain depth. This is easy to confirm in a human by comparing depth perception with one eye closed versus both eyes open, and has also been confirmed to exist in several animals; for instance Poggio et al[1] confirmed stereopsis in rhesus macaque monkeys using random dot stereograms. It is also easy to show that homo-sapiens make use of shape-from-shading. Ramachandran[2], details the concave/convex ambiguity, by which an object can, based on shading alone, be perceived as going either in or out, as demonstrated by fig. 1.1. Confirming this in animals proves to be tricky, but it does seem unlikely for it to be unique to our species. A creature that uses SfS must necessarily infer light source configuration as well, as SfS requires such knowledge to work[3]. From the above we can conclude that at least one evolved organism has the capability to use Stereopsis and SfS in an integrated system, indicating that such an approach has value.

The second reason focuses on how SfS and stereopsis complement each other - this follows on from the argument for the inclusion of LSE. Albedo estimation is under-constrained in areas with variable albedo - to estimate albedo requires multiple pixels with known relationships between their albedo, typically that of identical albedo. SfS therefore works best where albedo is constant and can be inferred. As previously stated, in constant albedo areas pattern matching fails and stereopsis does a bad job[4]. Alternatively, when the pattern is good and stereopsis works well albedo can not be inferred and SfS fails. To conclude, these approaches each do well when the other does not. Note the difference in the reasons for failure however - stereo fundamentally can not work in a smoothly shaded area, whilst SfS could work in a textured area if albedo were known.

Stereopsis and SfS also extract different types of detail. SfS provides surface orientation information, which is then integrated to get depth, whilst stereopsis obtains depth via triangulation. Surface orientation gives a lot of fine detail about a surface, but when integrated small errors are accumulated, resulting in large scale flaws in the output, referred to as curl. Correspondence on the other hand provides low resolution depth, missing details on the scale of single pixels due to the regularisation methods used. Small correspondence changes can equate

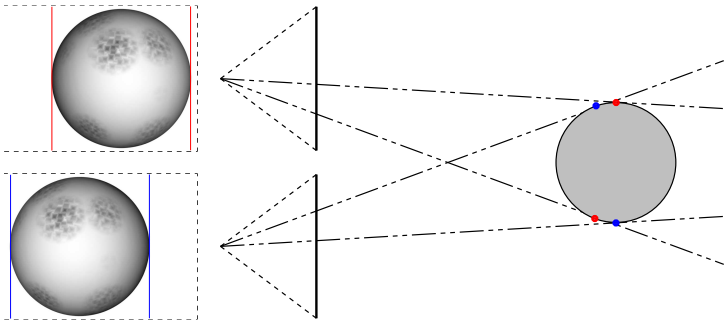


Figure 1.2: A scene that separately breaks both stereo and SfS; see text for details.

to large depth changes if the locations from which the images were captured are close together. To conclude SfS provides high frequency detail whilst stereopsis provides low frequency detail[5] - they again complement each other.

A simple example where either a stereopsis or SfS based algorithm would fail in isolation, but in collaboration could work, is a partially textured sphere, see figure 1.2. SfS will handle all non-textured regions, but fail with regards to the textured regions, whilst stereo will handle the textured areas and fail elsewhere. However, to make a further point, stereo will introduce a systematic error by matching the edges of the sphere, which SfS can not fix. As seen in figure 1.2 the occluding boundaries do not correspond.

Following this chapter is a literature review, which goes into much more detail for all three of the relevant areas. The first research chapter, chapter 3, gives an algorithm for combining shape-from-shading and stereopsis; it uses Gaussian belief propagation within a modular framework. Next is a belief propagation based shape-from-shading algorithm, given in chapter 4. It makes extensive use of directional statistics, using a distribution that has not been used with belief propagation before, which requires novel methods to handle. Chapter 5 finishes the research by describing a light source estimation algorithm. It uses an elegant probabilistic formulation which is optimised using branch & bound to find the optimal direction for a single point light source. The final chapter gives conclusions. Four appendices give background material; the first two cover the equations involved in taking a photo, specifically geometry and light transport. The remaining two appendices cover material used by the presented algorithms, in particular belief propagation and directional statistics.

Chapter 2
Literature Review

THE wide ranging nature of this work leads to a wide ranging literature review. Four sections constitute this chapter - shape-from-shading, stereopsis, combining them, and light source estimation. Later, in the appendices, ideas used by this work, rather than subjected to research, are also given a literature review.

2.1 Shape-from-Shading

Shape-from-Shading [SfS] algorithms solve the long standing problem of using the irradiance captured by a photo to infer the shape of a scene, as defined previously[1]. Originally introduced by Horn[6][2] solutions to this problem exploit a typical set of assumptions, specifically: Lambertian reflectance, orthographic projection, constant known albedo, a smooth surface, no surface inter-reflectance and a single infinitely distant and known light source. Despite using this set of assumptions the mapping from irradiance to surface orientation is many to one, and so there are a family of possible solutions[3]. Selecting a particular solution requires further information; this typically takes the form of a smoothness assumption. The following is divided into multiple sub-sections. First the SfS reviews are iterated, then algorithms are detailed and classified, before, finally, limitations are considered.

[1] Also referred to as photoclinometry, a term that is mostly reserved for inferring the topographical maps of moons/planets from satellite imagery, but is still used as a synonym for SfS.

[2] Earlier work exists, mostly concerning photometry of the moon. For instance Diggelen[7] observed that the moon was a Lambertian surface and inferred the relative depths of 1D slices rather than 2D areas. Working with characteristic slices and then combining them, sometimes even by hand, was typical prior to Horn. Rindfleisch[8] was probably the first to use a computer, in the modern sense of the word.

[3] This is often expressed as the irradiance constraining one degree of freedom when the surface orientation has two degrees of freedom. Whilst true for Horn's assumptions in general this is the best case scenario, unless an integration constraint is used - see 2.1.2.1

2.1.1 Reviews

In 1999 Zhang et al.[9] surveyed the area, concluding that Lee and Kuo[10] was the then state of the art. Lee and Kuo iteratively linearised the reflectance map and solved the resulting linear equation using the multigrid method. This approach is typical in defining a set of costs to minimise, specifically a cost for violating the Lambertian reflectance assumption and a cost for not being smooth; once defined the costs are minimised using a suitable method, in this case the multigrid method. In second place they put Zheng & Chellappa[11], which can be similarly broken down, though has a more sophisticated smoothing method based on equating the gradients between input and output. The difference in performance between these two algorithms can, for the most part, be put down to the solving method rather than the formulation. Additionally, both these methods work with depth rather than surface orientation, which has the advantages of enforcing integrability and negating the need for an integration step. This review paper also included Bichsel & Pentland[12], Lee & Rosenfeld[13], Pentland[14] and Tsai & Shah[15] in its testing. Synthetic inputs were standardised by this paper to the vase and Mozart inputs, which have ground truth. The error measurements used for comparison however have issues, discussed later in this subsection. Real world results were given, but without ground truth. Zhang et al.[9] ultimately conclude in their survey paper that SfS algorithms produce poor results.

Also provided by the 1999 review is a classification system for SfS algorithms, based on the solution method rather than the assumptions. As they are comparing algorithms the authors select a set that make the same assumptions, specifically the assumptions given at the start of 2.1.2, and so classification by this principally makes sense, despite fundamentally limiting the scope of the paper. The classifications are:

- *Minimisation*. An approach that minimises a cost function through some unspecified means.
- *Propagation*. An approach that propagates information from initial locations, such as singular points.

- *Local*. An approach that uses local information only; usually involves a classification of surface type.

- *Linear*. An approach that approximates the reflectance map by a linear equation, so it can be solved as such. Usually iterative with the approximation updated after each convergence.

This classification method appears somewhat dubious - for instance the first category can probably include all algorithms, as all algorithms are, implicitly or otherwise, minimising a cost function. Indeed, in the paper over half the algorithms are put into this first category, with a quarter in the propagation category and the remainder in the final two. It also tells you very little about the actual method to say it minimises a cost function, as there are many techniques for doing so. Many algorithms can fit into multiple categories too - for instance Lee & Kuo[10] is classified as being a minimisation approach, but it also linearises the reflectance map and so could be considered a linear approach. Each step of this algorithm is also local, and it propagates information from all pixels to all others by iteration - not precisely the meanings given by the authors, but very close regardless.

Dourou et al.[16] provide a 2008 SfS review. They observe that the qualitative best and quantitative best algorithms of the earlier review do not agree. No argument is given by them as to why, but to elaborate here it is an issue of *curl*. SfS algorithms are effectively collating surface orientation information, and small errors are inevitable. Integration then causes error to build up over distance, so you get large global errors, i.e. *curl*. This makes sums of squared depth error entirely inappropriate as large global errors will always overwhelm errors for fine detail, when fine detail is all that SfS can realistically extract with a high degree of accuracy. Unfortunately this second review, despite observing the problem, continues to use this error metric. The algorithms compared do not appear to be better than those of the earlier review, 'appear' being the operative word as the results are, perversely[4], not comparable with the earlier review. In addition no firm conclusions are given; in fact, little of note is provided by this paper.

[4]Same data set and error metric, different resolutions and light source directions.

2.1.2 Algorithms

Horn's original work gave a two step approach: first calculate surface orientation and then calculate depth, through integration of surface orientation. The concept of singular points is used, where a maximum in irradiance implies that the surface must be oriented directly towards the directional light source.

As an early SfS algorithm which exhibits many of the assumptions of later methods, Ikeuchi and Horn[17] obtain a needle map[5] by minimising an energy function consisting of the sum of two per-pixel costs. The brightness cost measures the error between the sensed irradiance and the irradiance calculated from the surface using the reflectance map. The smoothness cost measures the deviation from a smooth surface. Whilst proposed to provide extra information with which to resolve ambiguity the smoothness term dominates, and pulls the surface towards a plane. An iterative approach based on the calculus of variations is used to minimise the cost.

For initialisation this approach uses the boundary constraint - at the boundary of a *smooth* object surface orientation will be perpendicular to the viewing direction and normal to the edge. This constrains the boundary surface orientation to be known, in addition to orientation at singular points. Brooks and Horn[18] give a set of algorithms, with the first minimising the same cost function as Ikeuchi and Horn using a much simpler implementation. These algorithms exhibit a large number of assumptions/limitations:

- All needle maps are valid.
- Soft SfS constraint.
- Smooth surface.
- Lambertian shaded surface.
- Constant albedo/shading parameters.
- Orthographic projection.
- Single infinitely distant light source.
- No shadows, no inter-reflections.

Further algorithms will now be classified by their differences from the above list.

[5] A needle map is an assignment of surface orientation to each pixel in an image.

2.1.2.1 Needle map validity

Impossible surface orientations exist for arbitrary needle maps. The *integrability constraint* enforces that a surface can be integrated[6]. This can be thought of as the principle that if you start travelling and end up at the same place as where you started the sum of how far you travelled in every dimension must be zero. An arbitrary needle map does not necessarily satisfy this in the depth dimension.

Frankot & Chellappa[19] define the integrability constraint[7] as

$$\frac{\delta^2 z}{\delta x \delta y} = \frac{\delta^2 z}{\delta y \delta x} \tag{2.1}$$

where z is depth and x and y are image coordinates. They then formulate a method of projecting a needle map onto the *nearest* needle map that satisfies this constraint. This technique can therefore be used with any iterative algorithm - results are given for using this in collaboration with the Horn and Brooks[18] method.

Robles-Kelly[20] proposes a scheme to add an integration constraint by modifying the smoothing method in the Worthington & Hancock approach[21], which is covered in 2.1.2.2. This allows the use of a hard irradiance constraint, unlike the Frankot & Chellappa approach, making it one of the few algorithms to have both irradiance and integration as hard constraints. Unfortunately it iterates from an initial solution until it gets stuck in a local minima, so it's performance is more dependent on the initialisation than the algorithm. This curtails its competitiveness.

Any algorithm that solves for surface depth rather than surface orientation will enforce the integrability constraint. Leclerc and Bobick[4] solve the same cost function as Ikeuchi and Horn[17], but instead of solving for $\frac{\delta z}{\delta x}$ and $\frac{\delta z}{\delta y}$ solve for z, with central difference calculated differentials. A smoothness constraint is used as before, but it is reduced to zero as the algorithm converges, resulting in a smooth solution that minimises the brightness cost alone. Additionally they provide a variant that solves for albedo and light source direction, and suggest how to handle piece-wise constant albedo, though give no results for the later.

[6] A surface orientation vector field that satisfies the integrability constraint is *conservative/irrotational* in vector calculus terminology.

[7] This had been previously used, but earlier methods used it as a soft rather than hard constraint.

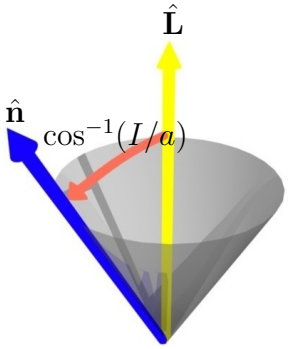


Figure 2.1: The geometric interpretation of the Lambertian reflectance information and how it constrains surface orientation to a cone.

Louw & Nicolls[22, 23] also solve for depth, taking a discrete approach that uses belief propagation [BP] (See appendix C). Unfortunately a discrete approach is unreasonable for this continuous problem. They attempt to compensate by running belief propagation repeatedly, each time with the same number of discrete depths for the algorithm to choose from, but with a smaller range of possible depth values assigned to those labels, selected based on the previous run. Despite using belief propagation for each step as a whole it fails to show the performance expected of such approaches. It gets stuck in local minima, due to the greedy nature of its search method - they report that this becomes a problem for inputs larger than 50x50. Heavy resource usage, complex factors and poor results also plague this method. Whilst a valiant attempt this method is ultimately doomed by it's discrete approach - a more intelligent approach is needed. This is provided in the next subsection, 2.1.2.2, by Potetz, who uses an effectively continuous distribution, as well as sophisticated optimisation methods to bring the resource usage mostly under control. Both of these methods are particularly relevant, as this thesis presents a BP based SfS algorithm in chapter 4.

2.1.2.2 Constraint hardness

A soft SfS constraint allows the output to diverge from the relationship that the input image should be generated by the output surface. Whilst this makes sense given noise and quantization artifacts the deviation is often unjustly far. This is most noticeable in algorithms such as Ikeuchi and Horn[17], where the smoothing term squashes the depth dimension.

Worthington and Hancock[21] observe that a Lambertian shading model causes the brightness constraint to limit surface orientation to a cone with axis angle θ, see fig. 2.1

$$\frac{I}{a} = \hat{n} \cdot \hat{s} = \cos(\theta) \tag{2.2}$$

They proposed a two step iterative algorithm - first apply a smoothness constraint; secondly project the normals back to their cones. Consequentially, the irradiance constraint is hard. An advantage of this approach is the smoothness component can be changed independently of the rest of the algorithm, this allows a variety of robust smoothing techniques to be tried and compared. Several improved versions of this work have since been published. Worthington has focused on using the algorithm to solve other problems, specifically re-illumination[24] and albedo estimation[25]. In the process improvements have been suggested, but no evidence is given that these result in any notable improvement.

The improvements of Robles-Kelly[20] to Worthington & Hancock have already been covered. An algorithm that shares the same use of hard constraints for both irradiance and integrability is given by Potetz[26]. It makes use of belief propagation [BP], which is covered in detail later (Appendix C.), but currently it serves to know that BP estimates the marginals of a multivariate probability distribution, often represented by a graphical model. Potetz makes use of two variables per pixel, $\delta x/\delta z$ and $\delta y/\delta z$, and uses various factor nodes to provide the reflectance information, smoothness assumption and integrability constraint. Linear constraint nodes, the focus of the paper, are an approximation method for reducing computation with clique sizes greater than two - time becomes a linear rather than exponential function of clique size. Variable width histograms are used to approximate the continuous probability distributions. These techniques result in a large speed up over a naive implementation, but resources are still an issue - memory consumption is at 6-10kB per pixel and runtime for a 128x128 pixel image is "several hours, although good results [...] appear after an hour or so."[27]. Whilst a resource hog this algorithm is probably the most capable available. Indeed, I expect that the Potetz result is near optimal, and future improvement will have to focus on computational efficiency, memory consumption and alternate models.

2.1.2.3 Surface family

Assuming a smooth surface is invalid for many real life scenes. Some robustness to discontinuities has been achieved by robust smoothing techniques - Worthington & Hancock[21] for instance. Zheng & Chellappa[11] provide a model where albedo and illuminant direction are additionally refined from initial estimates. Their relevant contribution however is in the smoothness term, which expresses the idea that when irradiance changes rapidly the image is probably less smooth, so the smoothness term should be reduced. This is implemented by preferring a solution where the change in image irradiance is equal to the change in calculated radiance, i.e. $\frac{\delta E}{\delta a} = \frac{\delta R(...)}{\delta a}, a \in x, y$. Presumably the explicit modelling of discontinuities in SfS would bring advantages, much as it does for stereo. Such algorithms could not be found in the literature however, though many authors claim robustness.

Some authors have avoided using smoothing alone to resolve ambiguity. Atick et al.[28] focus on statistical models of faces - they parametrise a general face with principal component analysis [PCA] and then use SfS to determine parameters for individual specimens. Dovgard and Basri[29] go one step further by also utilising the symmetry of the (typical) human face. Another example of this approach is the PhD thesis of Smith[30]. This also constructs a statistical model of a face, but does so in the directional domain of surface orientation instead of depth, which is a better fit to SfS methods. Initial work has distributions of direction for each pixel constructed in the tangent plane of the mean surface orientation using PCA, with only the first component, corresponding to a great circle, kept; more advanced techniques are also provided.

2.1.2.4 Non-Lambertian shading

The Lambertian Bidirectional Reflectance Distribution Function [BRDF] (See B.1.) used in the majority of algorithms is chosen out of simplicity and computational convenience. Whilst most objects show some Lambertian behaviour in reality few surfaces are actually Lambertian. Modelling specularities provides a significant improvement, with the advantage that there localised additive nature allows detection and subtraction prior to a traditional SfS algorithm, as an alternative to

using a complex shading model. Handling rim-lighting generally requires the use of a more advanced shading model however.

Several approaches have been published, but to give an example Healey & Binford[31] use the Torrence-Sparrow model (See B.3.1). Their approach is focused more on recovery of the model parameters based on detecting specularities - its specularity detection can fail however. It only solves for regions local to specularities and only offers 1D slices constructed assuming the surface orientation direction is constant as you move away from the brightest pixel. A more recent example of using the Torrence-Sparrow model is Bakshi & Yang[32], which modifies the algorithm of Vega & Yang[33]. The Vega & Yang algorithm[33] iteratively applies local heuristics to the surface normal at each pixel in a variational model; it is effectively a more sophisticated version of Brooks & Horn[18]. The modification then simply swaps the Lambertian shading model for the Torrence-Sparrow model as the model only ever needs to be evaluated for a given surface normal, as it relies on irradiance error alone with regards to using the irradiance information. Results are given for quadratic surfaces only however.

Ahmed & Farag[34] use a model other than the Torrence-Sparrow model, specifically the Oren-Nayar model[35] - this work is covered in 2.1.2.7 as it also considers falloff from the light source.

SfS information can assist with the detection of specularities; therefore integrating specularity detection will inevitably obtain improved results. Ragheb & Hancock[36] take such an approach using a Bayesian framework to weight pixels by their contributions from specular and Lambertian models. The weights and Lambertian surface normal (The specular surface normal is fixed as the bi-sector of the view direction and light direction.) are iteratively updated by smoothing the field of weighted mean normals till convergence.

As a final mention Lee & Kuo[37] give a version of their algorithm where they utilise the fact that the reflectance function is linearised at use, as that allows them to drop in an arbitrary differentiable reflectance function with minimal effort. Like every SfS algorithm the above require shading model parameters - the more sophisticated the model the more parameters required, and the greater a problem obtaining them can pose.

2.1.2.5 Constant shading parameters

Nearly all algorithms support variable shading parameters within the constraints of their model. For instance, most utilise a Lambertian model, so it is a simple adjustment to assign albedo on a pixel by pixel basis rather than for an entire image. The problem is in determining the albedo map from the irradiance map. Whilst algorithms exist to solely guess albedo (e.g. Tappen et al.[38]) an integrated approach is preferred. Fua & Leclerc[39] take such an approach when combining a SfS algorithm with stereo. It is an object centred approach that fits a 3D model of triangular facets to the data, minimising a cost function - they assign albedo as the average colour in facets, and use a cost to select for smoothly changing albedo between facets.

2.1.2.6 Camera model

The orthographic camera model is a reasonable approximation where the scene depth is small compared to the distance of the camera from the scene. It breaks down otherwise as cameras capture scenes using perspective projection rather than orthographic projection.

Lee & Kuo[10] solve under perspective projection, this necessitates solving for depth directly. Their algorithm requires as input the average depth of the scene, in addition to the other reflectance map parameters. All the object centered approaches deal with perspective (e.g. Samaras & Metaxas[40], Fua & Leclerc[39]), as do the viscosity approaches with light attenuation (e.g. Prados et al[41]).

2.1.2.7 Light model

The approaches so far have considered a single infinitely distant point light source; there are many other possibilities[42], most of which have not been explored within the SfS literature. This can be attributed to the dramatic increase in complexity, that comes with multiple and/or area lights. Such problems are often intractable.

Instead of a lighting model of greater complexity a simpler approach can be taken, by limiting the light to be emitted from the camera. This can be a reasonable assumption when using a flash for instance. Prados et al[41] solve this problem

with viscosity solutions, and also model lighting falloff. Using falloff removes the typical ambiguity and provides for impressive results, but consequentially the algorithm does not work if the falloff is not measurable. Primarily this requires the input object(s) be close to the camera. Ahmed and Farag[34] also use falloff information in a PDF framework, where they additionally use the Oren-Nayar model[35], rather than the typical Lambertian model.

Brooks and Horn[18], previously mentioned, incorporate a sky component into their algorithm to complement the infinite light source. This is principled on representing a diffuse sky as a hemi-spherical light and the sun as an infinite point light. They also talk about solving for an arbitrary reflectance map but no results are given.

Tian, Tsui, Yeung and Ma[43] propose a complex method for dealing with multiple light sources, including area light sources. A propagation approach is used, requiring that the depth of singular points be given.

2.1.2.8 Shadows and Inter-reflections

Both shadows and inter-reflections will distort the results of an algorithm that excludes these effects from its model. When handling shadows one solution is to use a shadow detection algorithm (e.g. Barnard & Finlayson[44]) and then remove detected areas from further calculation. Additionally, techniques based on deformable models, such as Samaras & Metaxas[40], can switch off shading costs for areas that are occluded from the light source. (Object centred approaches such as this also use steropsis; they are covered in detail in subsection 2.3.2.)

Nayar, Ikeuchi and Kanade[45] use an iterative approach to reduce inter-reflections. They reason that inter-reflections increase the irradiance of a scene above that of the basic lighting model, which makes concavities shallow when extracted by a Lambertian SfS technique. The solution is to apply a SfS algorithm as usual, take the result and calculate the inter-reflections. These will be less than the true inter-reflections due to the scene being shallow, so they can be subtracted to create an image with less inter-reflections. Repeat and the process should converge to the actual structure of the scene.

2.1.3 Limitations

It is clear that SfS provides limited information about the structure of a scene. Ramachandran[2] details the already mentioned concave/convex ambiguity, by which an object can, based on shading alone, be perceived as going either in or out. Figure 1.1 shows this graphically - the human mind can *flip* between the two modes of perception.

Belhumeur, Kriegman & Yuille[46] extended this further with the *bas-relief* ambiguity, so named after bas-relief sculptures which exhibit and use this ambiguity to their advantage. The bas-relief ambiguity applies to the shading of an orthographically viewed Lambertian surface. It allows for a skewing transformation of the object combined with an adjustment of the light source to be applied to a scene, such that from the camera position for which the bas-relief transformation was applied there is no change in the irradiance of the object. Furthermore, slight movements from this view point do not destroy the effect. The bas-relief ambiguity is a super-set of the concave/convex ambiguity for orthographic projection, but does not apply for perspective projection where only the concave/convex ambiguity is an issue. As the transformation also requires moving the light source it also only applies if the light source is also being estimated - if the true light source direction is provided then, again, only the concave/convex ambiguity applies.

To resolve ambiguity many algorithms[18, 21] initialise to a convex shape to prefer a convex solution over a concave solution. This works because they then get stuck in a local minima closer to a convex solution. The ambiguity no longer exists if attenuation of the light source is modelled - this does not make sense for an infinitely distant light source, but has been used with regards to a source at the camera[41].

2.2 Stereopsis

Stereopsis[8] uses two $2D$ projections of a $3D$ scene to recover the third dimension, depth. Ignoring the triangulation required to actually determine depth and taking advantage of the *epipolar constraint* this can be refined to the task of finding a $1D$ disparity function along each *epipolar line*, to match geometry between the images. See appendix A on two view geometry for the geometric details of this process. It now serves to give a precise definition of stereopsis considering geometric issues.

Input always includes a pair of *rectified* images, usually labelled left and right. Rectification transforms the images such that the epipolar lines match scan-lines, making the geometry of the captured images irrelevant to the stereopsis algorithm. Whilst usage is rare the function from pixel location to ray direction or the function from disparity to depth are both possible inputs. The rare usage is surprising considering they are necessary for correctly enforcing smoothness constraints and their value in tuning algorithm parameters to the input. Why this is the case is probably in part due to simplicity, but also the Middlebury stereo test[47, 48], which is popular for comparing algorithms, does not provide such information.

The output will be a *disparity* measure for each pixel, for one or both images. An indication of *unknown* is output by some algorithms[49], usually to indicate an occluded region. As the input is sampled the output is usually assumed sampled, however some algorithms output continuous representations[50]. Disparity can be best defined in terms of its relationship with the input. Given a $3D$ scene point imaged in each image as a $2D$ point the correct disparity for an imaged point will offset you to the imaged point in the other image. Due to rectification this is a 1D offset along a scan line.

Stereopsis is a very large field, with more papers than anyone could realistically read, let alone review. For this reason this review is mostly kept to a high level, discussing ideas with exemplar papers, though some specific papers are covered. Of the following three sub-sections the first reviews the surveys of stereopsis algorithms, the second explores a variety of current algorithms, whilst the final looks at approaches which have not fitted into the discussion up to that point.

[8] Often just referred to as stereo in computer vision literature.

2.2.1 Classification of approaches

Various surveys of the field of stereopsis algorithms have been published[51, 47, 52], all feature taxonomies of the (then) state of the art. They are now detailed in chronological order.

Dhond & Aggarwal[52] (1989) divide stereopsis into three steps, pre-processing (finding matchable features), establishing correspondence (matching features), and recovering depth. The final step can be considered solved. Steps one and two illustrate the structure of a sparse stereopsis algorithm. Key points are that no consistency is enforced between correspondences, and that no obvious model exists for interpolating the gaps between point/line features or filling in the matched areas of lines/regions.

Sparse stereopsis algorithms have since fallen out of fashion, with dense stereopsis algorithms the now preferred approach. Such approaches are the focus of Scharstein & Szeliski[47] (2002). In dense algorithms no feature selection stage occurs, instead the image is divided up into features with 100% coverage, and all are used. Usually single pixels are used, alternatively the segments from a segmentation algorithm[53] can be utilised[50]. This is advantageous, as such formulations can allow for consistency constraints between the now adjacent features, and no further interpolation is required[9]. In such approaches features are usually not localised or matched to sub-pixel accuracy, resulting in disparity maps with discrete steps. This leads to discrete steps in the resulting surface, unless a further post processing step refines the result.

Scharstein & Szeliski[47] divide algorithms into four parts, they then classify algorithms by their approach to each part. The parts are, in execution order:

1. **Matching cost computation**. Which correlation algorithm is used.

2. **Cost (support) aggregation**. How the correlation data is processed, usually the disparity space image is used directly, but schemes such as shift-able windows can also be used.

[9]This can prove to be a disadvantage however, as it can result in formulations using implicit interpolation without consideration of the consequences. For instance, a lot of earlier dense methods used squared disparity difference, which smoothed over and destroyed occluding boundaries.

3. **Disparity computation / optimisation**. The method of selecting disparities, the simplest is a winner takes all approach, but global approaches are preferred, such as those based on Markov random fields [MRF] and solved with dynamic programming or belief propagation.

4. **Disparity refinement**. A final refinement of the disparities to produce a better result. When it exists it is usually a continuous refinement with the previous step implemented as a discrete optimisation for efficiency reasons. Most algorithms do not have this step.

It can be observed that this is a classification of the solution method more so than the physical model being solved. The paper includes a quantitative comparison of algorithms, using a now well known test set. Since the original publication the rating system has been maintained and updated online[10], with most new stereopsis papers being submitted. At the time of writing, March '09, the top algorithm is Wang & Zheng[54], second is Klaus, Sormann & Karner[55] whilst third goes to Yang et al.[56] - these will be covered later. Results vary little between the top algorithms however, and if ranked by specific images from the test rather than a combined metric the ordering is very different for each image, with no algorithm showing consistency of placement.

In regards to the physical model there are many possible scene assumptions, yet there is little variation in application, with the majority of algorithms using the same formulation. This is a matter of practicality, as correspondence with irradiance provides insufficient information, necessitating solving for a limited physical model. The physical assumptions are now iterated, along with some algorithms that take the uncommon approach:

Solid objects. The uniqueness constraint[57] expresses that a feature in one view only matches with a single feature in any other view. Taking it further the two-way uniqueness constraint requires symmetry, i.e. if feature A is matched to feature B in one view then B must be matched to A in all other views. This assumes solid objects and fails under transparency. More

[10]The newest rankings can be found at `http://vision.middlebury.edu/stereo/eval/`, as of March '09.

critically, it fails when a single feature in one view is represented as two or more features in another view; this is most likely to happen with changes of scale between the two views.

Tsin et al.[58] explicitly consider transparency. The model gives each pixel a foreground colour and background colour, plus a mixing/alpha parameter. Foreground and background depth maps are calculated independently using graph cuts. The trick is in separating these layers in the first place, which may be done by minimising the difference between the predicted image with the current depth maps and the actual image - iteration is used. Results are poor, though this is expected given the difficulty of the problem.

A more conservative use of transparency can be found in Taguchi et al.[59] - they restrict transparency to the edges of segments, to model the fact that edge pixels are usually a mixture of colour from the two adjunct segments. The algorithm is otherwise a typical segmentation and plane fitting approach, though the planes are fronto-parallel and the segmentation updated as the algorithm is run, rather than done once to start with - this allows it to recover from an initial segmentation mistake.

Cohesive matter. The ordering constraint[60] observes that a point A that is left of a point B in one view will also be to the left in another rectified view. It fails in the case of a thin object suitably distant from a background, see figure 2.2. This is an expression of the idea that matter is generally clumped together into a few large objects, rather than lots of little objects.

Additionally, occlusion only happens at the edges of the matter clumps[57], making it relatively rare within most scenes. Algorithms can avoid modelling occlusion and accept poor performance in the region of occluding boundaries, however, the difference between algorithms that explicitly model occlusion and those that do not is dramatic. Modelling occlusion allows for sharp edges at object boundaries, algorithms that do not will tend to blur and/or poorly locate the edges. The cohesive matter argument also supports the smoothness constraint given below. Nearly all modern stereopsis algorithms explicitly handle occlusion.

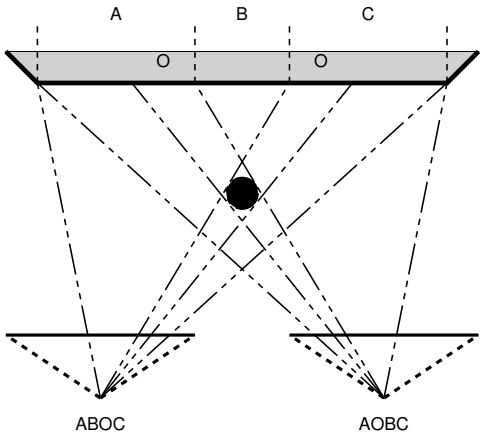


Figure 2.2: The example that breaks the ordering constraint. Note how the circle is on the right of background region B in one view but to the left in the other.

Textured surfaces / Smoothness constraint. In areas where insufficient information is present, i.e. areas without texture, a correspondence based approach fails. There are generally three approaches to this problem, namely to ignore it, detect and flag it or provide an interpolation capability. If you ignore it the results can be somewhat arbitrary, this is evident in the greedy correlation algorithms[11]. Detecting it is obviously better, but leaves gaps in the output which are perceptually ugly.

Alternately a smoothness cost[61] can be applied to provide a smooth interpolation in areas of insufficient information. This can smooth the entire result however, and cause problems at discontinuities. A correlation based approach can fail if given a pattern as it will match to the repetitions - a smoothness cost can resolve the ambiguity by preferring the smoothest choice.

Lambertian reflectance. With few exceptions[62, 58], the simplifying assumption of Lambertian reflectance is assumed. As a consequence a surface's recorded irradiance is independent of the viewer, providing a simple matching cost function in terms of colour differences. This becomes a problem with specularities, reflections and all other non-Lambertian effects.

[11]Greedy correlation works by calculating the correlation for a range of possible disparity values then using winner takes all, for each pixel. No consideration is given to consistency between pixels or views.

Planar surface model. Having a surface model to fit can dramatically reduce the information requirement and increase the robustness of an algorithm. For instance, if the input is of faces then providing a face model improves results. Of course, if such an approach is then given a turnip as input it is still going to output a human face, which would be entirely wrong. For general purpose stereo assuming a set of planes is popular[63], often with planes delineated by the results of a segmentation algorithm. This suffers from the problem just exemplified - the real world has non-planar surfaces.

Fronto-parallel planar surfaces. Window based correlation methods are assuming a fronto-parallel surface for the pixels within the window. This results in poor handling of surfaces that are at large angles to the camera and/or over occluding edges. Window based methods affect occluding edges by pulling them away from their actual position in the output disparity map, to make an object smaller or larger than it actually is. Many current algorithms match single pixels only, and have no problem; plus methods to improve the handling of both occlusion and large angles exist[64].

Orthographic projection. Algorithms mostly assume an orthographic projection rather than perspective projection, by using a uniform smoothing term for instance. However, this is not a noticeable problem unless applying stereo with fish-eyed equipment. Object centred approaches implicitly take this into account, see subsection 2.3.2.

Failure is not an option. Very few methods provide any form of explicit confidence measure, though some will indicate occluded regions[56] or insufficient information[49]. In fact, the Middlebury stereo test[47, 48] version 2 actually measures the capability of algorithms to *guess* disparity in areas of occlusion, where no correspondence information is available. Whilst this behaviour is reasonable for a stand alone module when integrating a stereopsis algorithm into a complete system it would be desirable for such a measure to exist. From a physical perspective this is a catch all, as inevitably the physical model will never be sufficient for all scenes.

returning to the review papers, Brown, Burschka & Hagar[51] classify by solution method rather than any breakdown of an algorithm's structure. They do however classify occlusion handling methods into three categories:

Detection. Detect and mark occluding regions, often as a post-processing step.

Sensitivity reduction. Use robust correlation methods, such as Zabih & Woodfill[65], to minimise the effect.

Geometric modelling. Integrate occlusion reasoning into the optimisation step.

Lin[66] goes further than classifying occlusion methods and defines a three axis classification scheme, it is:

Continuity Lin divides continuity into three possible approaches; constant, discrete and continuous. Constant considers all changes of disparity to be occluding boundaries whilst discrete allows for a smooth change. However, they both represent disparity using discrete values. Continuous allows for arbitrary real-valued disparities that smoothly vary. Note that whilst continuous is obviously preferred there are problems with solving such formulations, and doing so in a reasonable time frame.

Discontinuity Four classifications of discontinuity handling are provided; free, infinite, convex and non-convex. Free is taken to mean no constraint, whilst infinite prohibits discontinuities; neither of these models produce reasonable results. Convex means an ever increasing cost proportional to the size of the discontinuity, this produces sharp occluding boundaries. However, as the cost depends on the distance between an occluding boundary and the object behind that boundary the other costs can become overwhelmed. Non-convex overcomes the limitation on boundary depths by capping or otherwise damping the cost increase such that the size of an occluding boundaries disparity has no effect on the costing of that boundary.

Uniqueness Four categories are provided in regards to uniqueness; transparent, one way, asymmetric two-way and symmetric two-way. Transparent is no constraint, one-way requires that a feature in one image goes to one feature

in the other, but not that a feature in the other goes to one feature in the first image. Two-way's meaning is self-evident. An asymmetric two-way algorithm will handle the constraint differently in the reference image than the other image, whilst a symmetric approach will not.

In this sub-section several classification methods used by other authors have been detailed, and a physical formulation based classification provided. This physical model is weak with regards to stereo algorithms alone, as most are classified into only a few of the available buckets; however, when considering hybrid methods in the next section it's value will become apparent.

2.2.2 Highlighted methods

The previous sub-section considered a physical classification, with a few examples. Here classification is done by the solution technique, in much the same vein as Brown, Burschka & Hager[51].

2.2.2.1 Disparity Space Image

A 3D volume can be introduced for an image, indexed by disparity and pixel coordinates. Originally it was referred to as the spatio-disparity space image[67] - it is now usually abbreviated to just disparity space image [DSI]. This volume provides a measure of the matching cost between two images, for each possible disparity assignment, and is used almost universally by stereopsis algorithms.

DSI's are generally constructed using block-matching techniques, such as sum of squared differences, or normalised cross-correlation[47] [NCC]. Window based correlation metrics assume that the entire window is sensed in the other image with no transformation other than translation. If the window is at the occluding boundary of an object or there is a change in angle or scale between images then this assumption does not hold. Various methods have been proposed in regards to these problems.

One approach is to define a metric that is robust to errors, such as the Rank/Census approach of Zabih & Woodfill[65]. This considers the brightness ordering of pixels with regards to the pixel in the centre of the window, comparing these orderings

between windows. The advantage is much like that of taking the median rather than the mean - a few outliers will not be a problem; this method is also invariant to a change in brightness/contrast between images. Alternatively, a shifting window approach[68] can specifically handle borders by taking the best score, or most confident score, from a window in several different positions over each pixel. Changes in angle can be managed using an affine transform to transfer one window to the same coordinate frame as the other[64]. Changes in scale can be handled using a sampling invariant matching method, such as Birchfield & Tomasi[69, 70]. This uses linear interpolation between pixels to find the values of the half-pixels, it then considers the range of values available in pixel-sized square regions centred on each pixel. The cost between pixels is then the distance between the centre value of the pixel in one image and the closest point in the range in the other, this cost being zero if the pixels value is within the range - to make it symmetric they calculate this cost both ways and take the minimum. Szeliski & Scharstein[71] simplify this idea by taking the distance between ranges rather than a range and a point, which is of course zero if the ranges overlap. They also propose a stereo approach based on window matching, with various aggregation methods, to produce a dense disparity map without a global optimisation method.

Many global methods use a measure of the difference between two pixels rather than the difference between two blocks of pixels, relying on consistency constraints to find a reasonable solution. This has none of the problems of block matching methods, but provides less discriminative capability.

2.2.2.2 Local Approaches

Given a DSI the winner takes all approach can be applied to a subset of points that are *distinct* and *consistently selected* between views, where reliable matching may occur. The sparse nature makes consistency constraints difficult, so this approach can suffer badly from various physical assumptions and noise. Whilst many methods have been invented to improve the result of this simple approach[52] global methods can enforce significantly more advanced consistency constraints, and produce superior results.

When the epipolar constraint is unknown and hence rectification has not been applied you have a correlation problem with two degrees of freedom. In this case the sparse winner takes all approach makes sense, as a global approach would generally be intractable[12]. Using a corner detector[73] distinctive points can be selected. Robust methods such as RANSAC[74] can then be used to check for task-specific consistency. This is the approach used for fundamental matrix calculation[13], which is necessary for rectification prior to the application of a dense stereo algorithm.

2.2.2.3 Global Approaches

Many methods allow global consistency constraints to be expressed, e.g. dynamic programming[75], graph cuts[76] and belief propagation[77]. Whilst other techniques exist these three are the primary tools for solving stereopsis.

Dynamic programming finds the global minimum of a certain class of problem - it is limited to solving discrete Markov chains. For solving the stereopsis problem, e.g. Cox et al[78], it is not possible to optimise the entire image, so each scan line is separately processed. Consequentially a streaking affect forms, where individual scan lines do not line up; vertical consistency costs can be included to reduce this.

Graph cuts and belief propagation can both solve Markov random fields [MRF] with the *maximum a posteriori* estimator. Given a set of random nodes they are usually used to minimise an energy function of the form $E(f) = \sum_{x \in X} D_x(f_x) + \sum_{\{x,y\} \in N} V_{xy}(f_x, f_y)$ where X is a set of nodes and N is a set of pairs of neighbouring nodes [14]. The V function is preferred to be in various forms for speed[79], and is almost invariably a distance (metric) function. Typically discrete labels are used, f, representing a set of possible disparities, so D_x is the DSI and V_{xy} is the cost of changes in disparity - a smoothness term. Given the above functions these methods can find the labelling that maximises the joint probability function, $P(f) = \prod_{x \in X} \exp(-D_x(f_x)) \prod_{\{x,y\} \in N} \exp(-V_{xy}(f_x, f_y))$.

[12]For a slight movement between images, such as in video sequences, an optical flow[72] approach can be taken.

[13]The fundamental matrix expresses the relationship between two cameras. Given a point in one image it finds the epipolar line in the other; see appendix A.

[14]Triplets etc can be used, but the processing cost is prohibitive. When only using pairs it is referred to as a *pairwise* MRF.

Graph cuts solve the N label assignment problem using an optimal binary label swap technique[76], whilst belief propagation passes frequency functions as messages between nodes[15]. Tappen and Freeman[80] compare these two inference techniques, and find that graph cuts produce lower costs whilst belief propagation can produce faster results. They conclude however that there is no discernible difference between them, as both are producing significantly lower costs than ground truth for the stereo formulation they use, and the speed difference is not significant.

Belief propagation has significant advantages over graph cuts however, firstly it can also solve for the *minimum mean squared error* estimator. This solves for the mean marginal distribution of each random variable[80]. For discrete representations of continuous random variables, such as disparity, an interpolation scheme can be used where a parabola is fit around the label with largest probability and the continuous label that maximises probability selected - graph cuts lack the probabilistic output required to do this. This mostly removes the steps from the resulting disparity map.

Markov random fields are just one problem among many that can be solved using these methods. Bayesian nets and factor graphs, as well as many other models, can also be solved[77]. This allows a choice of representation convenient to the problem being tackled. Belief propagation works with continuous random variables[81], and methods for accelerating convergence have been found[79], making it an extremely powerful method.

A simple example of finding disparity using belief propagation may be found in Felzenszwalb & Huttenlocher[79]. They have a user set range of disparity values, and consider a labelling for each pixel, where each label corresponds to an integer disparity value in the range. The cost for a disparity value for each pixel is defined using a colourmetric distance between the matched pixels, whilst the cost of adjacent label assignments increases linearly in the disparity difference between them, up to a cap.

[15]Graph cuts produces an optimal result with a binary labelling, whilst belief propagation produces an optimal result if there are no loops in the graph. In all other cases the results are approximate, this is expected as the MRF problem is NP-hard. Belief propagation is often referred to as *loopy* belief propagation when solving for graphs with loops in.

Campbell et al.[49] are more sophisticated. Instead of considering all disparity values they only consider disparity values that are at peaks in a robust NCC score - this not only reduces processing but also reduces the blurring affect of regularisation. They also include an *unknown* term, set to a constant cost for selection, with terms to encourage unknown pixels to cluster. This unknown term covers a variety of cases, such as occluding boundaries, weakly textured surfaces and repeated texture. It also acts as a cost cap, limiting how high the cost of matching may go before it effectively gives up.

Xu & Jia[82] give a particularly intensive use of BP. Instead of modelling just one image from the stereo pair they solve for both images simultaneously. This requires linking each pixel in each image to all pixels in the other image that it could be matched to - these extra edges in the MRF make this quite a computationally expensive algorithm. Outliers are modelled in this technique using a soft outlier measure, assigned to each pixel, and hence BP is iteratively run as the outlier chance measure can not be updated at the same time as the disparity map. Segmentation and plane fitting can be used by this algorithm, by using disparity deviation from a fitted plane to influence the outlier measure - this is unusual as the final optimisation is per pixel rather than per segment.

2.2.2.4 Model Fitting Approaches

All approaches considered to this point have output disparity maps. The alternative is to output segments over which a particular function defines disparity[16]. The common choice is planes[50, 63, 66]. Such algorithms are structured into an initialisation step and a refinement step. The initialisation step for an automated stereo algorithm is likely to take the form of segmentation with the assignment of layers to each segment, boot strapping with a simpler stereo algorithm. Other initialisations are possible, including by hand[50].

The refinement steps in the literature use a variety of techniques, but an iterative greedy approach that optimises a single model property or single plane at a time is common. Optimising a cost function in terms of re-rendering is typical.

[16]The segments can overlap if transparency is modelled

This is constructed in terms of using the current model to re-render the input, often using forward or inverse warping[17], the cost then being the sum of absolute pixel differences between the two images.

2.2.3 Top Approaches

Three of the best algorithms, as defined by the Middlebury stereo test[47, 48][18], are now described. Results are not discussed - they all produce state of the art results, with little difference between them. This primarily serves the purpose of showing how many of the ideas given previously may be combined to create an actual algorithm; the top three methods are discussed in reverse ranking order.

2.2.3.1 #3: Yang et. al.

Yang et al.[56] give an iterative algorithm that involves both segmentation, plane fitting and per-pixel cost minimisation using belief propagation. They initialise by minimising a global cost function, involving a data term and a smoothness term. The data term uses a colour weighted correlation score - as pixels move away from the centre pixel, in terms of spatial and colour difference, they are down weighted, the idea being that this creates robustness to occluding boundaries where large colour changes are typical. Smoothness is absolute disparity difference, scaled to consider colour difference between the adjacent pixels. The cost minimising disparity map is solved for both images with belief propagation; for efficiency reasons it is done hierarchically, using lower resolutions to accelerate the convergence of higher resolutions and prune the search space of extremely unlikely matches.

Pixels are then categorised. First the left-right test is applied, i.e. it checks that the disparity for a left image pixel references a pixel in the right image that references back to the same left image pixel; if a pixel fails the test it is presumed occluded. Non-occluded pixels are then classified as *stable* or *unstable*, depending on the cost ratio between their highest and second highest match.

[17] Forward warping is distorting the left image using the left disparity to recreate the right image. Inverse warping is distorting the right image using the left disparity to recreate the left image.

[18] Taken from `http://vision.middlebury.edu/stereo/eval/` at the time of writing, March 2009.

Finally the third step is applied. This uses mean-shift segmentation[53] of the left image to define segments, in which planes are fitted to the stable pixels alone using RANSAC[74]. These planes define a new disparity map, which is then used to modify the original data term from the first step - an extra cost is added that will penalise matches far away from the segments plane by taking absolute difference; the cost multiplier depends on the occluded/unstable/stable classification of the pixel. The original belief propagation cost minimisation is then applied again, with this new data term, and this last step iterated with the new disparity map.

2.2.3.2 #2: Klaus, Sormann & Karner

Klaus, Sormann & Karner[55] propose a fairly typical plane fitting stereopsis algorithm, with assorted tricks to give it superior performance. The opportunity is taken to discuss the plane fitting approach in detail. Note that the just discussed algorithm of Yang et al.[56] was published later, and makes use of some of the techniques now discussed. A plane fitting algorithm first segments an image before assigning planes to each segment as a reasonable approximation of the disparity in each segment. The advantage is that each segment contains more information, improving reliability; such algorithms tend to do well at the Middlebury stereo test as they avoid mistakes and produce continuous answers, which gives them an advantage over the labelling approaches. Unfortunatly reality is not constructed as such - plane fitting produces good quantitative results that tend to be aesthetically poor.

When performing the first step, the segmentation, an over segmentation is preferred to avoid segments that are not the projections of a planar, or almost planar, surface. Mean shift[53] is a favourite approach, and the one taken by this algorithm. To assign planes to segments first reliable matches are found, usually using window based methods - the discussed algorithm uses two techniques, with a weighting parameter between them set to optimise the number of matches. The first technique is common - sum of absolute differences [SAD], the second is also SAD, but with colours multiplied by their gradients, calculated for each direction; the point of this is to improve robustness to lighting changes, as gradients are

independent of any additive effect. There will be many outliers in the matches - removing as many as possible from the match set is desirable. Deciding which matches are reliable can be done with a threshold, either on the cost directly or on the ratio between the highest and second highest cost. Left-right consistency is usually used, and in this algorithm's case is the only method taken.

Once each segment has a set of reliable matches a plane may be fitted to them. This plane will take the form $d = ax + by + c$, where d is disparity, x, y the pixel coordinates and a, b, c are the parameters to be estimated. Some algorithms apply simple least squares fitting[63], whilst the previous algorithm used RANSAC. The current algorithm calculates each of a, b and c separately, getting estimates by fitting each reliable pixel in each image slice and then applying a "Gaussian smoothed median" for robustness. With a set of segments, each with a disparity plane, a final optimisation step is usually run to find the best plane to assign to each segment. This is especially valuable for smaller segments that might have the same plane as other segments, but do not have enough pixels to reliably estimate it. Klaus et al. take a typical cost minimisation approach, defining the cost of assigning each plane to each segment using the original match costs, and also include a cost for adjacent segments being assigned different planes. They then minimise with belief propagation - each segment is a random variable, and every plane a potential label; the random variables of adjacent segments are connected by a factor that encourages identical plane assignment.

2.2.3.3 #1: Wang & Zheng

Wang & Zheng[54] is another plane fitting algorithm, so details will be kept brief. Segmentation is again mean shift. Initial disparity estimates use a correlation window where pixels from different segments are ignored. Plane fitting is performed with a 'voting' algorithm, that shares many features with the approach of Klaus et. al.[55] Each plane parameter is estimated separately - the slopes are estimated by taking all pairs of matches on each segment slice to get estimates, before constructing a histogram and smoothing with a Gaussian to localise the peak. The slope parameters are then used to get bias estimates for all pixels in

the segment, and the same voting procedure again used. This method is robust enough that no pruning step is required to remove bad matches.

Plane assignments are not updated, instead the individual planes are optimised with a cooperative optimisation method - this involves iteratively optimising regions of the entire cost function. The cost function contains a smoothness term that minimises the jump between adjacent segments; a data term that forward warps the segments to the right image and compares the colour; and an occlusion cost that detects occluded pixels between adjacent segments, which are then penalised.

2.2.4 Further approaches

The discussion so far has been limited to the case of two rectified images and the generation of a disparity representation, there are other configurations of interest.

Instead of a straight stereopsis algorithm one can consider a depth refinement algorithm. Such an algorithm will need initialisation, which could indeed be a stereopsis algorithm, but could also use another technique - a real time but low resolution laser range finder system is the example given by Yang et al.[83]. The point of such an approach is that this refinement can be continuous and high resolution, whilst the initialisation step can be coarse and low resolution, and hence faster. It can also be the *disparity refinement* step in Scharstein & Szeliski[47]'s taxonomy, for any stereopsis algorithm. Yang et al.[83] gives a successful example of such an algorithm. It creates a cost volume, limited to a fixed range per pixel around the discrete output of an initial stereopsis algorithm, before using a bilateral filter to consider support for each value. This filtering has sufficient discriminating capability over the small ranges used that selecting the best cost works. Being still discrete, though higher resolution, interpolation based on polynomial fitting is then used. Whilst simple the value of this algorithm is that when applied to the Middlebury stereo test[47] it, at the time of the paper's publication, improved the results of every single algorithm previously submitted and took the top spot[19].

When more than two images are used it becomes a *multi-baseline* technique,

[19]It is no longer the top algorithm due to more recent publications - it is unknown if using this refinement on these 'better' algorithms would again improve their scores.

a large field within itself for which only a taste will be given. If many views are involved the camera-centric reconstruction of disparity is often replaced with scene-centric representations, such as voxels or deformable models. Whilst combining multiple disparity maps to generate a consistent model of the scene will work it can not fully utilise the available information. Okutomi & Kanade[84] use a winner takes all technique with a SSD generated DSI. However, they transform the DSI from disparity space to inverse depth space, so that all pairs formed with a reference image are in the same coordinate system. They then sum each pairs inverse depth DSI. The reasoning is that errors in each individual DSI caused by bad matches will change position between the various pairs, whilst the correct match will not, therefore the correct matches will line up and produce the best scores.

Voxel methods such as space carving[85] and voxel colouring[86] attempt to label a set of voxels as belonging to the object or not. Space carving is related to the visual hull approach. The visual hull approach constructs a volume which must contain the scene using silhouettes of multiple views as boundary constraints on the volume. Space carving uses radiometric constraints to create a shape which is visually consistent with each view. Given a hull which contains the scene, this can be calculated using the visual hull technique or another approach, space carving removes voxels on the surface that are not visually consistent until no bad voxels remain. This leaves the *photo hull*, the largest object that is visually consistent with all provided photos. The real scene must be a subset of this volume. Voxel colouring requires an "occlusion compatible norm", that is a camera configuration for which you can visit each voxel in a sequence, knowing that all voxels that could occlude the current voxel in any image have already been visited. In consequence the scene can not intersect the convex volume of the camera locations. Given this ordering a surface is found with an image consistent colouring. This approach is extremely efficient, but requires accurately calibrated camera configurations of a certain form. It also handles areas without texture badly.

Unlike voxel methods with their high memory requirements deformable models[39] only represent the current surface, iteratively deforming it to reduce a cost function. Whilst this significantly reduces memory requirements such tech-

niques will get trapped in the local minima of their cost functions very easily, and so must be initialised close to their true shape.

So far we have considered algorithms without consideration of runtime. There are many real world scenarios for the use of stereo where speed matters, such as the various robots on other planets where a human operator would prove useless due to the round trip communication time. Stereo is also of value to interactive applications, such as augmented reality - these and other scenarios require a soft real time algorithm. Such an algorithm is evidently not going to do as well as the algorithms already considered, as it has less resources to use. Indeed, many real time algorithms have been obtained by simply running older algorithms on faster hardware, and/or working with low resolution input - many do not even have a global optimisation procedure. Most modern techniques make use of a graphics processing unit [GPU] implementation, as GPU's are now cheap commodity hardware for solving massively parallel problems. For instance, Wang et al.[87] perform dynamic programming on the CPU whilst using a GPU for cost aggregation. The dynamic programming is entirely standard, the aggregation considers a vertical window to reduce streaking whilst keeping the computational cost low, and down weights pixels that are far away and have a very different colour from the central pixel.

2.3 Shape-from-Shading & Stereopsis

In this section the few hybrids of SfS and steropsis that exist are explained. They have been loosely classified into three sub-sections.

As a historical aside, such hybrids were first considered in detail by Blake, Zisserman & Knowles[88]. Whilst they considered many ideas no algorithms were provided. Their main thrust was aimed at the concept of using SfS to interpolate a dense surface between the sparse correspondences of a (sparse) stereopsis algorithm. Mostafa, Yamany & Farag[89], covered in 2.3.1, implement this exact idea.

2.3.1 Modular approach

A modular approach is one that runs stereopsis and SfS separately then attempts to combine the results. Leclerc & Bobick[4], in their previously mentioned SfS algorithm, use stereopsis to bootstrap their height field before applying there SfS algorithm. The stereo is used to provide boundary conditions and an initialisation close to the ground truth, to avoid the local minima their SfS algorithm gets stuck in. Note that this scheme has no robustness to either algorithm failing, and will get it wrong where only one algorithm has behaved correctly.

Hougen & Ahuja[90] work with the reflectance map rather than any particular model of light source(s) and BRDF(s) (See appendix B). Their technique estimates stereopsis once, and then iterates between SfS to determine shape and modelling the reflectance map. The reflectance map is modelled using two costs, one to enforce smoothness, and another based on the difference from actual irradiance given the current estimated shape. Two segmentation modules are used, one based on colour and another based on depth, reliable segmentation is assumed where the segments coincide, and unreliable when they do not. Unreliable areas are then ignored for subsequent use of the stereo inferred depth map. The depth map is directly used to initialise the SfS module, and indirectly used to initialise the reflectance map which is then used by SfS. The final result is output by SfS only. Additionally, in Hougen's PhD thesis[91], several earlier iterations of this algorithm are proposed.

Mostafa, Yamany & Farag[89] take the results from both modules and fit a smooth surface to {*stereo depth* minus *SfS*}, then add *SfS* back to the difference surface. The stereopsis module outputs sparse results, with only the sparse points being used for the surface fitting. Consequentially a dense estimate is output, which is the SfS output distorted to approximately match the sparse stereopsis data.

Cryer, Tsai & Shah[5] observe that SfS can be associated with high frequency information, as in the local details; whilst stereo can be associated with low frequency information, as in the general shape of the scene. This view is justified in the introduction of this work. Their technique consists of applying the two approaches separately then adding them in the frequency domain, using a filter loosely based on the human visual system to limit the frequency range used for each algorithm accordingly. The paper compares the results of each module on its own with those of the combined results, and shows considerable improvement.

2.3.2 Object refinement approach

Object refinement is used to refer to an object centred approach, i.e. an algorithm that uses the 3D model of the object being recovered directly. The model is initialised using the result of stereopsis alone, then both stereopsis and SfS are used to refine the model. Such approaches consistently suffer from being trapped by local minima - the initialisation is very important. Whilst three approaches are now described the differences between them are minor enough that this category really represents a single, but well explored, algorithm rather than a class of algorithms.

Fua & Leclerc[39] represents the scene as a regular triangular mesh with albedo information, they then use multi-baseline stereopsis and SfS plus a decreasing smoothness cost to iteratively refine the model. The overall cost of a given solution is the sum of several local properties for each triangle:

- Surface deviation from being a planer and equally spaced mesh.
- Correlation between the surface and where it projects onto the images. The 3D model is used to zero this cost when occlusion occurs.

- Deviation of calculated albedo from constant.

Albedo is calculated using the SfS constraint from the known surface orientation, average image irradiance and light source direction. The correlation algorithm is designed to provide no information in smooth areas whilst the scale of the albedo smoothing cost is directly related to the smoothness of the irradiance. This results in a scheme where SfS takes priority in smoothly shaded areas and stereopsis takes priority in textured areas. The formulation is optimised using a conjugate-gradient descent procedure, noting that initial optimisation is done with SfS switched off to allow boundary conditions to form.

Samaras, Metaxas, Fua & Leclerc[92] re-implemented the algorithm of the last paragraph five years later. Unlike the earlier version this implementation presumes piece-wise constant albedo and uses information theory to perform the segmentation. Additionally, it is applied to faces only and hence a face model is fit to stereopsis results for initialisation - this mostly overcomes the problem of getting stuck in a local minima. Specularity is identified and the light source direction is refined. The result is a capable face reconstruction algorithm. Samaras & Metaxas[40, 93] had previously published an algorithm of similar scope to Fua & Leclerc's algorithm, hence the collaboration.

Fassold, Danzl, Schindler & Bischof[94] takes, by the authors own admission, the exact same form as the previous paragraphs algorithm, except applied to archaeological objects rather than faces, so it reverts to model-less initialisation by stereopsis. To quote "The main differences between their and our approach are different optimisation methods, a different energy function and a different discretization of the object."[94]. Most of these changes affect an increase in speed rather than that of quality, but a multi-resolution approach is utilised such that the grid eventually obtains pixel-scale resolution.

2.3.3 Single formulation approach

A single formulation approach minimises a single cost function that includes terms relating to both lighting and correspondence.

Shao, Chellappa & Simchony[95] suggest a SfS algorithm similar to Leclerc &

Bobick[4] with the same depth map initialisation technique, they then suggest an extension of this SfS algorithm that integrates with stereo. No results are given. They minimise a cost function, and extend the basic SfS algorithm from three terms to four terms. Minimising three variables, the two partial differentials of depth and depth itself, the original costs are

- squared difference between SfS predicted irradiance and actual irradiance;
- squared difference between surface partial differentials as stored and as calculated from depth map;
- and a smoothness cost, sum of squared surface second differentials.

The extra term to integrate stereopsis is a cost for the squared difference between the SfS predicted irradiance in the left image and the actual irradiance at the correct position given depth in the *right* image. Cost is minimised using a gradient descent method, which requires a good initialisation to work.

Jin, Yezzi & Soatto[96] assume the image has already been divided into two sets - areas of constant albedo suitable for SfS and areas of texture suitable for stereo. It then applies a different cost function to each and minimises using level sets. Whilst they formulate the algorithm with consideration for specularities their results are limited to synthetic Lambertian scenes only.

2.4 Light Source Estimation

Estimating light source information is a fairly wide ranging problem with many possible inputs and many possible outputs that can be combined with many other tasks. In this work we are primarily concerned with estimating a single infinitely distant light source as input to a SfS algorithm, and so our focus is on algorithms designed to do just this. We continue to review algorithms with more sophisticated and alternate models however as they can contribute to solving the simpler case. The following subsections give key algorithms in this area, categorised into suitable groupings in approximate chronological order. We also review some algorithms for extracting just albedo.

2.4.1 Image Only

Pentland[97] gave a pioneering approach to light source estimation. Input is only a single image with no further shape or albedo information. This is an incredibly ill-posed problem, and so the assumptions of isotropic geometry with Lambertian reflectance lit by a single infinitely distant light source are used. The premise is the idea that given an isotropic object, i.e. an object which is smooth and fully contained in the image; you can then compare the image to a sphere lit from various directions. Light source direction is then selected identical to the best matching sphere. Best match is defined in terms of statistics on the differentials of irradiance. Interestingly, in competing with humans the estimator made some of the same mistakes. Chojnacki, Brooks & Gibbins[98] observed a resolution dependence with Pentland's estimator, which they then solved.

Yang & Yuille[3] solve for multiple light sources using boundaries and singular points. At the boundaries of objects the surface orientation is known, assuming a smooth surface. By collecting these points and performing a Hough transform they can solve for the x and y component of the light sources. Using singular points they can then solve for the z component. It requires that the lights be 'separable' to solve for the z component, and lights that are too near to each other will be merged when solving for x and y - it cannot handle more than 3 lights in practise. The light vectors in this system are presumed multiplied by the albedo

and light source strength, so albedo is also inferred. A key contribution of this paper is the introduction of *virtual light sources*. This is the observation that if you map all surface points of a Lambertian surface onto a Gauss sphere then segment by the shadow lines of each light source, i.e. where the sphere normal is 90° from a light source direction, then each segment can be modelled as lit by a single light source. In segments where there is only one real light then it is an actual light, but when multiple lights illuminate a segment it is a virtual light constructed by the sum of non-shadowed real light vectors.

Another example of boundary use is Nillius & Eklundh[99]. Their actual estimation technique is almost identical to Yang & Yuille[3], except for the inclusion of an ambient term. Output is only a single light source and they offer no method for getting the z component. They do however make the approach robust and usable for cluttered real world scenes. Firstly they devise a method to find potential occluding boundaries of smooth objects - they use an edge detector with chaining to get potential edges and then filter for sections with little change in shading, no other edges interfering, and with smooth direction changes. They again use linear algebra to extract light sources from these boundaries, but instead of using the boundary directly make use of interpolation from further into the object to avoid rim-lighting effects. To combine the results from the many edges found in an image they then use a Bayesian network, which includes random variables for estimates being wrong; to do so they calculate distributions on everything, including the answer. Real world results are presented, and show superb results given the complexity of the input.

Zheng & Chellappa[11] offer two light source estimation methods as well as an albedo and SfS estimation method in their extensive paper. The first light source estimation method assumes the image to be made of locally circular patches which then vote for the best light source direction. Solution two uses object boundaries - see Brooks & Horn[18] below for an earlier example of boundary use that also handles more than one light source. After estimating the light source direction they estimate albedo and then apply SfS. Real results are given.

Kim et al.[100] use a *bump* detection procedure. They find maximal irradiance points in the image and posit that each is at the centre of an elliptical region with

an ellipsoid-related surface patch. The patch is fitted to the irradiance in the region giving them known geometry, and allowing light source direction to be inferred. Results show it working well if it can find suitable regions, but it obviously only works for the subset of images with the desired properties. It also fails to resolve the concave/convex ambiguity and so provides two answers. Compared to a boundary based method this has the advantage of processing regions, which improves robustness.

2.4.2 SfS with Light Source Estimation

Our motivation for finding the light source is to pass it to a SfS algorithm. Several authors have integrated SfS and light source estimation into a single framework. This is motivated by each method being tractable if given the output from the other. Brooks & Horn[18] iterate between solving for shape and solving for the light source, which includes the (constant) albedo. They design a cost function in terms of satisfying the SfS constraint, having a smooth surface and a Lagrange multiplier for unit length normals. Assuming the light source fixed gives an iterative solution for surface orientation whilst assuming the surface orientation fixed gives an equation for the light source direction. Initialisation is potentially an issue and limited testing was performed, but they give an error of 2.7° for a sphere.

Samaras & Metaxas[101] use deformable models to do SfS as well as a least squared minimisation for the estimation of a single light source in an iterative approach. This comes with the advantages of being a detailed model - shadows can be compensated for, perspective is considered, any differentiable reflectance model can be used. The disadvantage, universal to deformable models, is the need for a very good initialisation as it will stop at the first local minima. They do reduce a stiffness parameter as the model runs in an attempt to avoid this. Results are given and appear to be qualitatively good, but there appears to be a weakness when lighting is head on. This is possibly a consequence of different parts of the model heading for different solutions and neither solution becoming dominant.

2.4.3 Known Geometry

Given known geometry you evidently know the surface orientation of every pixel in your image. This makes the problem not only tractable but over-constrained, such that it is practical to solve for sophisticated models, for instance models with many lights or non-Lambertian shading. An example of the later is Ikeuchi & Sato[102] who solve for the Torrence-Sparrow model (See B.3.1). They use a two step process, the first of which calculates the albedo and light source direction, the second of which calculates the specularity strength and sharpness. In this first step they iteratively use least square fitting to calculate values, but for each iteration re-segments the image. The segmentation is into Lambertian areas, specular areas and shadowed areas. Further iterations only use the Lambertian areas and so the algorithm effectively uses each estimate to prune bad data and make the next iteration a better estimate. The second step, specular parameters, also uses least squares, iterating between each parameter. Extensive results are given, including for human heads.

Zhang & Yang[103, 104] detect multiple directional light sources from an image of a perspective-projected constant-albedo Lambertian sphere. They introduce critical points, which are points on the sphere where a light stops having influence, i.e. points where the surface normal is perpendicular to a light source direction. These are the points on the boundaries of the regions which define virtual light sources, as introduced by Yang & Yuille[3]. By providing a robust procedure to find these critical points and then grouping them by great circle they allow the light directions to be determined, up to a sign, i.e. the recovered direction can be 180° from the actual direction. Once the directions are determined linear algebra recovers strengths and resolves the sign using irradiance values on the sphere. This method is later extended by Wang & Samaras[105] to arbitrary geometry, see next subsection, 2.4.4. Bouganis & Brookes[106] solve the same problem, again by detecting critical points. Their critical point detector is much simpler than Zhang & Yang[103, 104], and it handles three cases where Zhang & Yang fail. Specifically, it detects light emitting from the camera, which will produce no critical points, simply by inserting such a light into the final optimisation and deleting it again if

its strength is optimised as zero. They also handle cases where the light source directions are not linearly separable by selecting arbitrarily from the set of simplest solutions, and handle antipodal lights, i.e. lights with opposite directions. Another example of using a Lambertian sphere is Ortiz & Oliver[107] who only solve for a single light with an ambient term; they are much more robust however, if simply due to the simpler model. They have a particularly detailed noise model also.

To continue a theme of calibration from spheres Powell et al.[108] use reflective spheres. The method for directional lights is rather obvious as finding the specularities is a simple local maxima finding problem, but they consider lights of finite distance from the scene. They handle this by using multiple reflective balls and triangulating. Because they have to calibrate for the sphere positions they actually use spheres with one hemisphere reflective and the other Lambertian - reflective for triangulation, Lambertian for finding the edges. They are mounted to be turned without changing position.

The previous algorithm does not solve for relative light source intensities, whilst Zhou & Kambhamettu[109] do when assuming a calibrated stereo pair of a single sphere with both Lambertian and specular shading. Using the sphere edges they calibrate its position and radius. They then use a model with ambient, Lambertian and specular lighting, where ambient and Lambertian do not change with the view whilst specular does - this allows the specular regions to be separated from the rest by thresholding. Specular information is used to provide light source directions, with the averages of the centroids from each of the stereo pairs used for robustness. Lambertian information then constrains the relative light source intensities and the ambient term. For synthetic input the algorithm has an average direction error of 0.3° and an average intensity error of 3%; for real input they only give average direction error, at 3°.

Zhou & Kambhamettu also[110] provide a method for handling area light sources. They again use a sphere and find the specularities, ray trace them to intercept a plane and assume an area light source covering the area where the rays hit. Whilst simple this assumes that the specularity thresholding does not get pixels that do not point at the area light source, an assumption that seems unreasonable. Finally, Zhou & Kambhamettu[111] provide a method to handle a

variety of light source types, effectively combining the works covered in this and the previous paragraph. It is however the approach given above applied twice, to each image of the stereo pair. The regions for the same light in different images are then matched by adjusting the depth of the plane - this adds distance from the scene to the area lights, and allows them to end up as approximate point light sources if the distance is such that the region scales to be small.

Weber & Cipolla[112] solve for a single point light source, i.e. a light source that is not at infinity, and also include support for a spotlight with simple falloff. The camera is a perspective rather than orthographic one. A Lambertian object of known shape is used in the scene - in their testing they use a cube. They initialise with a linear model where the light is at infinity before switching to the non-linear model with the finite distance light - it is solved with the Gauss-Newton method. Extensive and reliable results are given, but only for a cube in a blacked out room, a rather limited and arguably easy case.

Lagger & Fua[113] give an algorithm that tracks specular regions between multiple images of a textured scene of *moving* rigid objects. The movement is key, as texture moves with the object whilst specularities move depending on the shape of the object and position of the light sources. Assuming that shape is available for each image of the scene (e.g. as calculated using shape from motion.) they can then calculate multiple light source directions. They also recover an albedo map for the objects in question, i.e. the image without the specularities and diffuse lighting effects, as well as parameters for the Ward lighting model (See B.3.2). The first step is to detect specularities in each image, they do this by growing regions around maxima which suppress lesser maxima and then filtering so only regions brighter than their neighbours remain. Consistency is then enforced between images and light source directions voted for, before its all refined in an optimisation framework that infers lighting parameters, including colour. This final optimisation also adds and estimates an ambient term; it makes use of RANSAC[74]. Real world results are shown, including an example case of augmented reality.

Hara, Nishino & Ikeuchi[114] solve for a single point light source rather than a distant light source and also calculate the parameters of a Torrance-Sparrow model (See B.3.1). They initially describe a system where the alternate between

extracting the diffuse information for estimating the light source position and using the remainder once diffuse information is subtracted to estimate the specular parameters. The next iteration around the better specular parameters allow for a better removal of specular affects. This approach cannot handle textured surfaces however, so a second approach is proposed. It uses the natural logarithm to make a simplified Torrance-Sparrow model linear and then uses correlation to select a good model from a subset of models constrained by the specularity extents. Unfortunately it requires as input a specular component image, unlike the first, and is, by the authors own admission, less robust.

In a further work Hara, Nishino & Ikeuchi also[115, 116] approach the problem using directional statistics (See appendix D); they also solve for specularity parameters as well as light source directions and relative strengths. They first introduce the Spherical Torrance-Sparrow model (See B.3.1), where they have substituted the Gaussian distribution in the specular term with a directional distribution, the von-Mises-Fisher distribution; this has little actual affect on the shape of the distribution and is done for mathematical convenience. If one considers a ray travelling from the camera to the scene, striking an object and reflecting in a given direction we may construct an *illumination sphere* of scalars where each point is the recorded brightness of the object for that reflection direction. This only works under the assumption of constant albedo and known shape. Using this idea they represent the illumination sphere as a mixture of von-Mises-Fisher [vMF] distributions, and fit it using an expectation-maximisation framework. Due to the construction each VMF distribution in the fitted mixture then corresponds directly to a light in the scene, having matching directions, and the (shared) concentration parameter of all the distributions maps to the surface roughness whilst the relative weights in the mixture map to the relative strengths of the light sources.

2.4.4 Shadows

Shadows provide lighting information if you know the geometry of the casting objects and receiving objects. Sato, Sato & Ikeuchi[117] propose a scheme of determining a lighting field with a known object on a known surface, in their

examples a cuboid on a flat plane. The lighting field is constructed as a set of infinitely distant point light sources in the directions of vertexes on a geodesic dome. For each point in the scene they then work out what percentage of each point light will illuminate that pixel - 0 if it is in shadow or $[0, 1]$ otherwise, depending on the points BRDF and orientation. This results in a linear equation to solve for the light field. A known BRDF is required for this approach to work, but they also give a solution for unknown BRDF where you assume it is Lambertian and calculate the albedo by taking a photo before you introduce the shadow casting object, which of course assumes constant albedo. Either approach is practically an active vision approach, or requires human involvement, due to the need to know the shape of the occluded parts as well as the visible parts of the shadow casting object(s). Later work introduced the simultaneous estimation of the parameters of the Torrence-Sparrow model (See B.3.1), as well as the lighting field, via an iterative approach so BRDF is not required; and also used an adaptive lighting field[118]. Another followup paper[119] improved robustness using three techniques. The first considers the robustness of solving for the current lighting field and simplifies it if the linear equation is too close to singular. The second treats the point light sources as regions for shadow calculation, allowing the algorithm to correctly handle penumbras without a dense lighting field. Finally, the third weights the pixels so having a large number of pixels with one lighting arrangement does not overly bias the approach for satisfying their constraint.

Wang & Samaras[105, 120] take the works of Zhang & Yang[103] and Sato, Sato & Ikeuchi[117] and combine them. That is, they combine a shading method and a shadow based method. The shadow method remains untouched but the shading method is extended. They use a refined version of Zhang & Yang[103] for initialisation but then refine the results using the shading within virtual light regions as it is more reliable. A system for removing spurious lights and finding the real light sources from the virtual lights that result from the refinement step is also used. Shadows are integrated partially to prune spurious results early in the algorithm, to save time, but primarily to add possible critical boundaries, to find lights that shading alone would have missed. This is of some value as instead of a calibration sphere they can use objects of arbitrary but known shape, mapped onto

a sphere. Sampling gaps may exist on this sphere, which would be problematic if using the shading approach alone. Later work[121, 122] extended the above further, to handle specularities by detecting and deleting them and to represent low frequency area light sources via spherical harmonics. The low frequency area light sources were posited as an explanation for the remaining error once directional light sources were estimated.

Li et al.[123] integrates shading, shadows and specularities into a single framework that is also robust to texture as long as sufficient constant albedo areas remain. They sample light source directions from a hemisphere, for each direction they calculate expected shadow boundaries, specularities and critical points. The image itself is processed to find potential examples of these cues and the extracted result and simulated results are compared. Light source directions that show considerable overlap are kept as long as the overlap exists for at least two cues and is better than nearby alternate light source directions. The value in this approach can be seen for example in the shadow edge estimation - they use a simple edge detector which also finds texture edges, but by simulating and comparing texture edges are ignored. The same applies to critical points and specularities. Detailed and particularly impressive results are given. The same limitation of known objects where the self occluded parts are known applies as much to this work as the other shadow-using works however.

There are several approaches that represent light source directions using a method such as spherical harmonics, and can be considered as continuous extensions of the discrete approach of Sato, Sato & Ikeuchi[117]. An example is Okabe, Sato & Sato[124], which compares spherical harmonics on a sphere with Haar wavelets on the faces of a cube. The spherical harmonics represent the light source strength for the light source direction which matches the surface normal on the sphere. In using only a low number of harmonics they produce a linear equation to estimate the harmonic weights. Using a frequency analysis they show, not unsurprisingly, that this approach cannot handle point light sources or area light sources with sharp edges as they are high frequency and therefore ignored by the small set of harmonics used. They also show that occluded shadows actually reduce the potential resolution of the result. Given these results they then use

Haar wavelets applied to each face of a cube. Light source strength is then found by firing a ray from the centre of the cube in a given direction and sampling the strength at that point. When solving they use a hierarchical scheme which adapts the resolution of the wavelets to the available data, avoiding the shadow problem.

2.4.5 Shading Only

In addition to reviewing light source estimation algorithms that sometimes estimate albedo it serves to also briefly review algorithms that estimate shading only[20]. Such an algorithm is of value to SfS as extracting shading alone from an image allows the typical assumption of SfS - constant albedo - to be compensated for when the input has variable albedo. Funt, Drew & Brockington[125] introduce the main concepts of such an algorithm. They move to a luminance/chromaticity colour space, specifically $l = R+G+B$ for luminance and $r = R/l$ and $g = G/l$ for chromaticity. It can be observed that the image is the multiplication of the shading (reflectance) image and the colour (albedo/intrinsic) image, a relation that becomes additive if you take the logarithm. They take the gradients of the logarithm of the luminance image. Under the assumption that large chromaticity changes correspond to changes in colour whilst small changes correspond to changes in shading they threshold the chromaticity images and where the threshold is exceeded they zero the gradients of the $\log(l)$ image. This new gradient image is then integrated, with care taken to handle the resulting curl, and converted back to a luminance image which represents the shading alone. This algorithm depends on colour only for edge detection, and so can not detect changes of colour that are purely brightness, i.e. dark blue to light blue; the integration also means that a multiplicative constant is lost. A gradual change of colour would be interpreted as a shading change.

Weiss[126] takes the idea of the derivative of the log image being sparse in a different direction. Motivated by webcams they tackle the problem of extracting a single intrinsic image from a set of frames where the intrinsic image is roughly constant over all frames but the reflectance image is varying between frames. A

[20]Which can be albedo for Lambertian objects, and is often referred to as the intrinsic image, as opposed to the reflectance image, which contains lighting information only.

particularly clever and effective method of removing illumination effects is given for this scenario, involving the medians of results from a filter bank, which is shown to be the maximum likelihood result under fairly weak assumptions on image statistics.

To give an example of a more recent algorithm Tappen et al.[38] use exactly the same principles, copying parts of both of the above algorithms. Their contribution is a better method for identifying which derivatives are caused by shading and which are caused by colour changes. Specifically, as well as colour they also use changes in brightness to detect colour changes, using an Ada Boost based classifier. They then propagate information using generalised belief propagation[127] (Appendix C) to get consistent selection along edges. The given results show it to be exceptionally effective.

Chapter 3
Combining Shape-from-Shading & Stereopsis

COMBINING shape-from-shading [SfS] and stereopsis is the express purpose of this thesis - we now tackle this problem directly. Key to providing an algorithmic solution is overcoming the many differences between these two techniques, which make combining them in a single optimisation scheme particularly difficult. For this reason we instead propose an algorithm for combining the results of these approaches, before iteratively rerunning the SfS component using the extra information gleaned from combining. There is also an issue of albedo estimation, as mentioned in chapter 1. Variable albedo is required if stereo is to produce much meaningful output, but variable albedo is a problem for SfS. No reasonable choice exists other than to also solve for albedo.

The algorithm proposed is modular, with a SfS module, a stereopsis module and an albedo estimation module, plus this chapters primary contribution - a module to combine them. Gaussian belief propagation provides the optimisation framework for this module. There are also secondary contributions to be found in combining the modules. Next we have a section that extends this introduction and goes into further depth on the reasoning behind the system. After that details of Gaussian belief propagation are provided, along with how it is used to combine stereo and SfS results, ready for the following section on the actual modular algorithm. Finally results with analysis are given, followed by a conclusion.

3.1 Rationale

Chapter 1 already covered in some detail why it is desirable to solve this problem - here we are concerned with justifying the specific solution given. Key is the formula relating disparity and surface orientation, specifically if $d(x,y) \in \Re$ is disparity at a given pixel and $\hat{\mathbf{n}}(x,y) \in \{x \in \Re^3; |x| = 1\}$ surface orientation at a specific pixel then, assuming planar rectification, the relation is

$$\hat{\mathbf{n}}(x,y) \propto \left[\frac{bf}{d(x+1,y)+n} - \frac{bf}{d(x,y)+n}, \frac{bf}{d(x,y+1)+n} - \frac{bf}{d(x,y)+n}, p_d \right]^T \quad (3.1)$$

where two view geometry, specifically equation A.10, has been used to convert disparity to depth. The distance between adjacent pixels is represented by p_d, which is actually a function of disparity if perspective projection is used[1]. Surface orientation, $\hat{\mathbf{n}}(x,y)$, also needs a SfS constraint to be applied, e.g. the Lambertian reflectance cone constraint (See B.2 in the appendices, specifically equation B.6), another non-linear equation:

$$I(x,y) = a\hat{\mathbf{n}}(x,y) \cdot \hat{\mathbf{l}} \quad (3.2)$$

Combining the approaches in a single framework therefore has to include the above as constraints; not only that but to get the fine detail of SfS it must be a continuous optimisation. The constraint has to be applied to every pixel, which will also have matching costs and smoothing constraints applied. We have not considered albedo estimation in this discussion, which increases complexity yet again. Current techniques find this optimisation problem difficult if not impossible to solve, so an approximation/simplification is made.

Several different simplifications can be made, as considered by the literature review in section 2.3. We run the algorithms separately - this makes the relationship something computed for fixed values outside any single optimisation step. Apart from being tractable this approach has the advantage of modularity, so different stereopsis, SfS and albedo estimation algorithms can be easily experimented with.

[1]This equation is not actually used by the implementation - that triangulates into 3D space, which is much more complicated, but removes the *planar* rectification assumption and supports perspective projection.

There are three shape representations we could optimise with - disparity, depth and surface orientation, as we can convert from any one to another. Surface orientation is excluded as converting from it to the others can be problematic due to curl. Disparity was chosen over depth - the errors typical of stereopsis are best modelled in disparity space as we have regular match cost measures for each pixel - if converted to depth then these measures become unevenly spaced out and the pixels of variable size.

For the presented framework we use the Gaussian distribution rather than anything more sophisticated. Speed is a major advantage of this, as the Gaussian distribution is simple and requires no approximation; the disadvantage is that the information from neither stereopsis or SfS is actually a Gaussian distribution. However, taking the cost function around the lowest cost peak of a stereopsis algorithm and fitting a Gaussian distribution is not unreasonable if we assume that this lowest peak is close to the actual answer. Effectively we are assuming that we need to refine the accurate but low resolution/discrete stereopsis output to account for the sub-pixel resolution information provided by SfS. The probability distribution for the SfS is problematic, so much so that we just assume a fixed Gaussian around its point estimate; it is also assumed that dz/dx and dz/dy are not related, which is not true. Regardless, the experiments conducted (See section 3.4) give reasonable results, despite these simplifications. Given the above, this presented approach can be considered a stereopsis refinement algorithm, as mentioned in subsection 2.2.4 - the difference is that instead of refining using a better model it instead refines using an entirely different model, specifically SfS. It can be observed that any source of orientation information could be used with the presented method, and whilst not investigated it could, for instance, be used to combine stereopsis and shape from texture.

3.2 Gaussian Belief Propagation

Belief propagation [BP] in general is discussed in appendix C; in this chapter it is used specifically to find the posterior distribution of a Markov random field [MRF]. Gaussian belief propagation [GBP] is used to refer to belief propagation where the distributions passed as messages are represented by a single Gaussian. For this section belief propagation will be formulated on a Markov random field rather than a factor graph, as done in appendix C. Using MRFs instead of factor graphs is convenient from a representational point of view, but also has optimisation advantages as messages can be passed directly between variables - this reduces the message storage requirements and halves memory consumption. The use of the chequerboard update pattern[2] from Felzenszwalb & Huttenlocher[79] then halves it again.

Below first discusses the actual use of GBP, for combining stereo and SfS, and the graphical model constructed to solve this problem; following that the next subsection details the algorithm in abstract mathematical terms, which is then followed by specific implementation detail. Finally the section ends with some analysis of what this approach is actually doing.

3.2.1 Integration

As already stated, the task is to combine stereo information and SfS information in the disparity domain. We construct a grid shaped MRF, see figure 3.1, where each random variable represents a pixel's disparity. In this formulation we need two items of data - the prior on each pixel, which is derived from stereopsis, and the relationship between adjacent pixels, which is obtained from SfS. Given this information belief propagation can find an approximation of the optimal disparity map that combines both sources of information.

Stereopsis provides a direct estimate for the disparity at each pixel - the only

[2]This is where half the nodes are updated in one iteration then the other half in the next. A chequerboard pattern with each colour indicating which update set a node belongs to necessarily forms on a square grid by the requirement that each node being calculated only depend on nodes not being updated in the same iteration. This means that instead of incoming and outgoing messages for each pixel we only need to store the incoming, as they won't be updated when the pixel is using them as its neighbours are not updated at the same time.

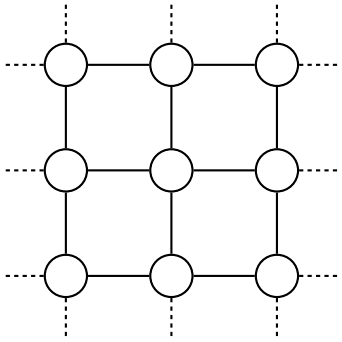


Figure 3.1: A Markov random field in a grid configuration, the configuration used by the integration algorithm. Each circle represents a random variable on the disparity of a pixel. Each edge represents the relationship between two adjacent pixels, in terms of disparity difference.

problem is to assign a standard deviation to its estimate, which is covered in the next section. This provides the Gaussian prior for each pixel. SfS on the other hand provides the differential of depth, where depth is inverse disparity - a complex relation as previously discussed. Each pixel is assigned a disparity from stereo and a surface orientation from SfS, therefore two view geometry (Appendix A) can be used to convert the disparity to a point in 3D space, which in combination with the surface normal defines a plane. This plane can then be intercepted with the rays of adjacent pixels (A 4-way neighbourhood is used.) to get an estimated depth, which may then be converted back to a disparity value, see figure 3.2. Taking the difference between the disparity value of the pixel in question and the estimated disparity of the adjacent pixel gives a disparity delta. Using this delta the relationship between adjacent pixels is an expected difference, represented by a Gaussian. The standard deviation of this Gaussian is set constant for the entire image, which completes the relationship between adjacent pixels. SfS and stereopsis therefore influence the MRF in different ways - the pixel prior models the depth information whilst the compatibility between sites is used to incorporate the orientation information.

Disparity deltas are calculated relative to the stereopsis or previous iteration provided disparity value. Because of perspective as the disparity is changed the delta should also change; the algorithm does not however do this due to the associated computational cost. This is effectively assuming a specific orthographic

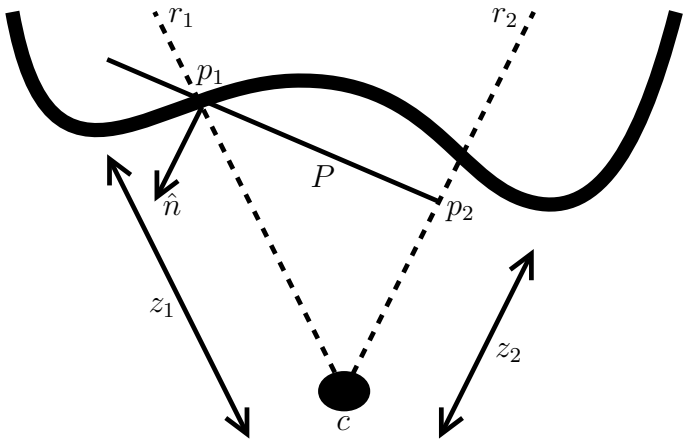


Figure 3.2: 2D representation of the calculation of disparity difference values. Given a disparity value for pixel one we can calculate its depth, z_1. Camera calibration is available, which gives a ray in space r_1, starting at the camera, c. This provides a point in space, p_1. SfS provides the surface orientation at this pixel, $\hat{n}$, which when combined with the point in space defines a plane, P. This plane can be intercepted with the ray of a second pixel, r_2, to give an estimated position for that pixel, p_2. It is then a simple matter to convert the position to depth, and then the depth to disparity, from which a disparity difference can be calculated.

projection for each pixel. However, as the change should not be too great and the algorithm uses this module iteratively, reinitialising the differences each use, no problem should, nor is seen to, occur. As a further note this technique can be simply adapted to be a SfS integration algorithm. By working with depth and calculating depth differences alone the SfS surface orientation map is converted into a depth map, with curl removed. (An orthographic projection is used.) The depth prior for each pixel has its standard deviation set to infinity, indicating no information, except for one pixel, which is set to be some arbitrary value to stop the system choosing a potentially numerically extreme depth basis. This integration (In the calculus sense.) method is used in the next chapter, 4, to visualise the SfS output.

3.2.2 Message Passing

We now detail the equations of GBP, in a Markov random field framework. Loopy belief propagation on a MRF works by iteratively passing messages between random variables. The message that node t passes to its neighbour, s, at time step

n is[128]

$$M_{t\to s}^{(n)}(d_s) \propto \int_{d_t} \psi_{st}(d_s, d_t)\psi_t(d_t) \prod_{u\in N(t)-s} M_{u\to t}^{(n-1)}(d_t)dd_t \tag{3.3}$$

d_t is the disparity at node t; $\psi_{st}(d_s, d_t)$ is the compatibility distribution between the disparities at t and s, and is obtained from SfS; $\psi_t(d_t)$ is the distribution of disparities inferred from the observed evidence, which, in this case, is obtained from the output of stereopsis. $M_{u\to t}^{(n-1)}(d_t)$ is a message from the previous iteration; and the set $N(t) - s$ is the neighbourhood of t excluding s. We can then compute the belief at node t using

$$B_t^{(n)}(d_t) \propto \psi_t(d_t) \prod_{u\in N(t)} M_{u\to t}^{(n-1)}(d_t) \tag{3.4}$$

As all component terms are Gaussian the messages and final belief are also Gaussian.

The prior and joint distribution between adjacent disparities need to be defined. $\psi_t(d_t)$ is a simple Gaussian distribution

$$\psi_t(d_t) \propto \exp\left(\frac{-(d_t - \mu_t)^2}{2\sigma_t^2}\right) \tag{3.5}$$

where μ_t is the expected disparity value, taken as the disparity estimate provided by stereopsis, and σ_t is the standard deviation, the calculation of which is given in subsection 3.3.1. $\psi_{st}(d_s, d_t)$ is the relationship between adjacent pixels disparity values, d_s and d_t. For completeness when defining it we include a multiplicative factor, in addition to the additive factor we obtain from the SfS information. Specifically we use

$$d_s = m_{st}d_t + u_{st} + N(\sigma_{st}) \tag{3.6}$$

In the current use the multiplicative factor, m_{st} is always set to 1, so we only have a simple offset, u_{st}. The offset is provided by the SfS information - surface orientation provides a depth difference between adjacent pixels, which can be converted to a disparity difference. $N(\sigma)$ is the noise term, specifically a zero mean Gaussian with a standard deviation of σ_{st}. We therefore determine the joint

probability distribution between adjacent disparities as

$$\psi_{st}(d_s, d_t) \propto \exp\left(\frac{-(m_{st}d_t + u_{st} - d_s)^2}{2\sigma_{st}^2}\right) \tag{3.7}$$

which is of course another Gaussian distribution. Note that proportionality has been used above, and will continue to be used. This is because the normalising constant is irrelevant during actual calculation, and can be calculated once done.

3.2.3 Implementation

Further to the above the actual implementation details are given, in terms of simple operations. Using a variant of the Gaussian algebra of Cowell[129] the Gaussian distribution is defined as

$$\phi[\mathbf{P}\mu, \mathbf{P}](\mathbf{x}) = \phi[\Sigma^{-1}\mu, \Sigma^{-1}](\mathbf{x}) \propto \exp\left[-\frac{1}{2}(\mathbf{x} - \mu)^T\Sigma^{-1}(\mathbf{x} - \mu)\right] \tag{3.8}$$

$\mathbf{P}$, the inverse covariance matrix, is referred to as the precision, and $(\mathbf{x})$ generally dropped for readability. The reason for defining ϕ in this way is that it produces simple rules for manipulation; specifically multiplication is given by

$$\phi[\mathbf{P}_a\mu_a, \mathbf{P}_a]\phi[\mathbf{P}_b\mu_b, \mathbf{P}_b] = \phi[\mathbf{P}_a\mu_a + \mathbf{P}_b\mu_b, \mathbf{P}_a + \mathbf{P}_b] \tag{3.9}$$

If we add an additional *independent* variable to a distribution (extension), we get

$$\mathrm{Ext}(\phi[\mathbf{P}\mu, \mathbf{P}]) = \phi\left[\begin{pmatrix} \mathbf{P}\mu \\ 0 \end{pmatrix}, \begin{pmatrix} \mathbf{P} & 0 \\ 0 & 0 \end{pmatrix}\right] \tag{3.10}$$

Given that

$$\mathbf{P} = \begin{pmatrix} \mathbf{P}_{11} & \mathbf{P}_{12} \\ \mathbf{P}_{12}^T & \mathbf{P}_{22} \end{pmatrix} \qquad \mathbf{P}\mu = \begin{pmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \end{pmatrix} \tag{3.11}$$

marginalisation over the first set of variables is

$$\mathrm{Marg}_1(\phi[\mathbf{P}\mu, \mathbf{P}]) = \phi[\mathbf{h}_2 - \mathbf{P}_{12}\mathbf{P}_{11}^{-1}\mathbf{h}_1, \mathbf{P}_{22} - \mathbf{P}_{12}\mathbf{P}_{11}^{-1}\mathbf{P}_{12}^T] \tag{3.12}$$

We begin from equation 3.3, the message passing equation. Using the fact that everything is Gaussian, including the messages, we define

$$M_{t\to s}(d_s) = \phi[P_{t\to s}\mu_{t\to s}, P_{t\to s}] \tag{3.13}$$

Using the rules for multiplication, equation 3.9, we can then break up the message passing equation as

$$M^{(n)}_{t\to s}(d_s) \propto \int_{d_t} \psi_{st}(d_s, d_t)T(d_t)dd_t \tag{3.14}$$

where

$$\begin{aligned} T(d_t) &= \psi_t(d_t) \prod_{u\in N(t)-s} M^{(n-1)}_{u\to t}(d_t) \\ &= \phi[P_t\mu_t, P_t] \prod_{u\in N(t)-s} \phi[P_{u\to t}\mu_{u\to t}, P_{u\to t}] \\ &= \phi\left[P_t\mu_t + \sum_{u\in N/s} P_{u\to t}\mu_{u\to t}, P_t + \sum_{u\in N(t)-s} P_{u\to t}\right] \\ &= \phi[P_0\mu_0, P_0] \end{aligned} \tag{3.15}$$

where $P_0\mu_0$ and P_0 are introduced for convenience, being defined as $P_0\mu_0 = P_t\mu_t + \sum_{u\in N/s} P_{u\to t}\mu_{u\to t}$ and $P_0 = P_t + \sum_{u\in N(t)-s} P_{u\to t}$. P_t and μ_t come from the pixels prior,

$$\psi_t(d_t) = \phi[P_t\mu_t, P_t] \tag{3.16}$$

We then take $T(d_t)$ and extend it using equation 3.10, which gives

$$\mathrm{Ext}(T(d_t)) = \phi\left[\begin{pmatrix} P_0\mu_0 \\ 0 \end{pmatrix}, \begin{pmatrix} P_0 & 0 \\ 0 & 0 \end{pmatrix}\right] \tag{3.17}$$

Now $T(d_t)$ is extended it may be multiplied by $\psi_{st}(d_s, d_t)$, which is defined from manipulating equation 3.7 as

$$\psi_{st}(d_s, d_t) = \phi\left[\frac{1}{2\sigma^2_{st}}\begin{pmatrix} -m_{st}u_{st} \\ u_{st} \end{pmatrix}, \frac{1}{2\sigma^2_{st}}\begin{pmatrix} m^2_{st} & -m_{st} \\ -m_{st} & 1 \end{pmatrix}\right] \tag{3.18}$$

therefore the result of the multiplication is

$$\psi_{st}(d_s, d_t)\text{Ext}(T(d_t)) = \phi \left[\begin{pmatrix} P_0\mu_0 - rm_{st}u_{st} \\ ru_{st} \end{pmatrix}, \begin{pmatrix} P_0 + rm_{st}^2 & -rm_{st} \\ -rm_{st} & r \end{pmatrix} \right] \quad (3.19)$$

where we have defined $r = 1/2\sigma_{st}^2$ Finally, we marginalise away d_t to leave d_s, using equation 3.12. This provides the actual message

$$M_{t\to s}^{(n)}(d_s) = \phi \left[ru_{st} + \frac{rm_{st}(P_0\mu_0 - rm_{st}u_{st})}{P_0 + rm_{st}^2}, r - \frac{r^2 m_{st}^2}{P_0 + rm_{st}^2} \right] \quad (3.20)$$

except in the case of the problem we are solving $m_{st} = 1$, so

$$M_{t\to s}^{(n)}(d_s) = \phi \left[ru_{st} + \frac{rP_0\mu_0 - r^2 u_{st}}{P_0 + r}, r - \frac{r^2}{P_0 + r} \right] \quad (3.21)$$

We iteratively apply the above message calculation rule to find an estimate of the MAP disparity field. Finally, once convergence has occurred, the beliefs are given by equation 3.4, which is explicitly

$$B_t^{(n)}(d_t) = \phi \left[P_t\mu_t + \sum_{u\in N(t)} P_{u\to t}\mu_{u\to t}, P_t + \sum_{u\in N(t)} P_{u\to t} \right] \quad (3.22)$$

The final step is to calculate the estimated disparity

$$\mu_t = \frac{P_t\mu_t + \sum_{u\in N(t)} P_{u\to t}\mu_{u\to t}}{P_t + \sum_{u\in N(t)} P_{u\to t}} \quad (3.23)$$

One advantage of this scheme is the use of inverse covariance, precision. This allows a standard deviation of ∞ to be set, indicating that no information is known about the disparity of a given pixel. In the case where the stereopsis algorithm indicates certain pixels are unknown, due to occlusion etc., this is done. Of course, this works both ways as the final belief of a pixel could be a precision of zero, making the above equation 3.23 undefined. For convergence to actually occur relationships between pixels have to be balanced, that is if pixel s thinks pixel t should differ in disparity by δ then pixel t should think that s differs by $-\delta$. To enforce this the average of the absolute difference is used, with the correct signs.

3.2.4 Equivalence with linear methods

Recent work[3] has, not that unsurprisingly, shown that GBP has an equivalence with solving linear equations. Bickson, Shental et al.[130] use GBP to solve the typical linear equation, i.e find $\mathbf{x}$ for $\mathbf{Ax} = \mathbf{b}$, for the purpose of decoding a noisy communication channel[4]. In the process they show that GBP converges faster than typical methods, such as the Jacobi and Gauss-Seidel algorithms, which are shown to actually be specific variants of GBP[131], so whilst it can be concluded that the preceding approach has a linear interpretation the solution method used is probably the best available. GBP is also much better suited to specifying and reasoning about the problem at hand, as encoding the given problem as a set of linear equations with variable standard deviations would prove messy. The same authors, Shental, Bickson et al.[131], in a followup paper show that max-sum and sum-product GBP are *identical*, so whilst we have referred to our approach as finding the marginals for each pixel these happen to also be the most likely estimate for all pixels.

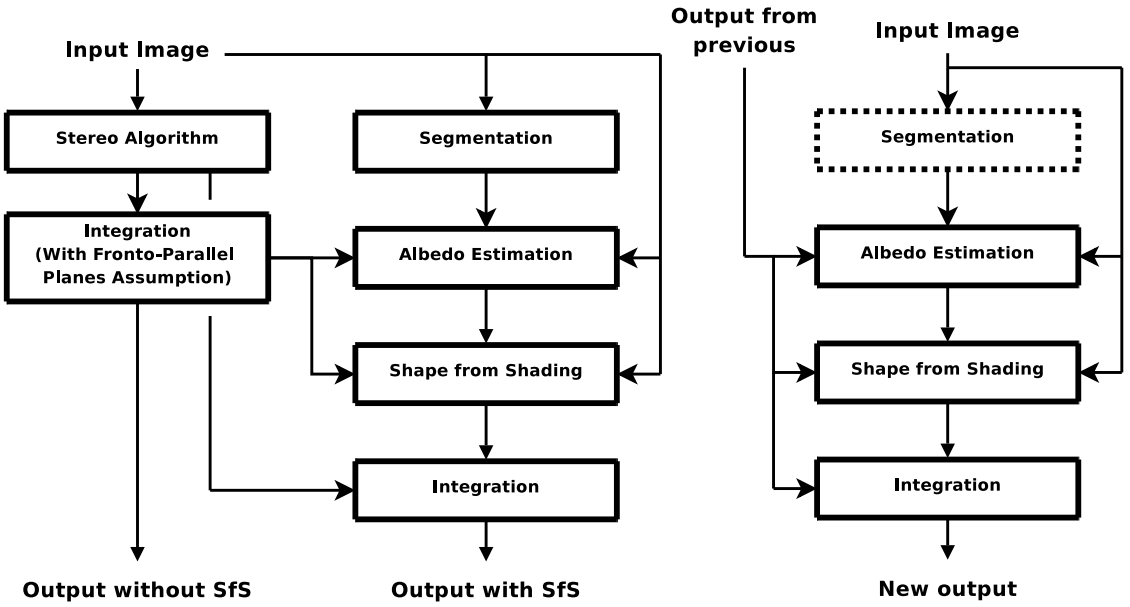
3.2.5 A wider view

So far the presentation has focused entirely on a stereopsis derived depth estimate and a SfS derived orientation estimate. Whilst some of the given details are specific to these inputs most are not, and there is no reason to limit the approach to stereopsis and SfS exclusively - other sources of input may be used. A related work in this regard is Nehab et al.[132], which replaces stereopsis with an active variant, and uses photometric stereo to determine surface orientation. The active stereopsis introduces a projector to the standard stereo arrangement, capturing different projections of light onto the scene over time - this allows each pixel to have an unique feature vector within its epipolar line, which makes the correspondence problem trivial. This is an accurate technique, far more so than typical stereopsis;

[3] Published around and after the publication of the first paper this chapter is originally derived from.

[4] It should be noted that GBP is an embarrassingly parallel problem, and the formulation such that zero entries in the $\mathbf{A}$ matrix can be effectively ignored. The most valuable contribution of this paper in the authors estimation is therefore a relatively easy method for solving large linear equations with an *unstructured* sparse matrix.

additionally they over-determine photometric stereo using five light sources, to get very accurate normals. The actual algorithm itself consists of two parts - first a correction of orientation information using position information, then a correction of position information using orientation information. Correcting orientation is done be replacing the low frequency orientation component with the low frequency orientation of the stereopsis; this is identical in principal to Cryer, Tsai & Shah[5], the only differences lying in implementation. Some concern exists here as they treat normal vectors as points in 3D space for smoothing, before renormalising, which is both incorrect and strange, given that the correct solution is trivial. The second step combines the corrected orientation with the positions, and is based on solving large sparse matrices to find a surface that minimises squared distance from both the given positions and the tangent vectors calculated from surface orientation. This is mathematically similar to the presented approach, though it works with depth rather than disparity and gives no consideration to the error variability per-pixel of the given estimates. Unsurprisingly the results are much better than the presented algorithm, on account of the better inputs, but if given the inputs used by the presented it is expected that it would give worse results, due to the limited consideration of input error.



(a) Bootstrap (b) Iteration

Figure 3.3: Main data flows between the modules of the complete system. On the left we have the basic algorithm, on the right an iteration, which can be applied to the output of the left and then to itself repeatedly. The right hand diagram has segmentation in a dotted box to indicate that it is not actually rerun, but reused as its output does not change. See text and subsections for descriptions of each module; the integration module is described in the previous section.

3.3 Algorithm

Previously, in section 3.2, the integration module of the presented system was discussed. How this module fits into the complete system, and the other modules of the system, is now considered. The following discusses figure 3.3, which documents the modules and how they fit together. First consider the bootstrap step. It starts by applying the stereopsis algorithm to get an initial disparity map. The problem with this initial algorithm is that it produces discrete disparity estimates, so surface orientation calculated from these estimates is quantised to only a handful of values. This make this disparity map entirely useless for albedo estimation, so we smooth it. Smoothing is done using the integration algorithm that is core to this method, run identically to the final integration run of the bootstrap, except instead of a SfS provided needle map it takes as input a flat needle map, where surface orientation always points in one direction, towards the viewer. Output from this step, whilst created to support the albedo estimation,

is also used for comparative purposes in the experiments section as it shows precisely the difference between using SfS and not, at least for the bootstrap step.

Albedo estimation assumes segments of constant albedo - this is not very realistic for textured areas, but in such areas stereopsis results should have a small standard deviation anyway, so SfS should not matter much. As input it requires a segmentation algorithm - we use mean shift[53]. Using the smoothed disparity map to obtain surface orientation and the input image[5] to obtain irradiance each pixel provides an albedo estimate - these are then combined to get an albedo map. The albedo map is used by SfS to parametrise the Lambertian shading model used, SfS hence outputs a surface orientation map. Note that for this algorithm we are assuming the light source to be a known infinitely distant point light source. SfS uses a disparity map as initialisation, to bias it towards selecting between concave/convex interpretations that match the stereopsis output. Finally, the integration module from above is used to combine this needle map and the original discrete disparity map, outputting a refined disparity map.

Given the above description the details of the iteration step are self-explanatory - the refined output from the bootstrap is simply refined again, with as many iterations as desired. This refinement process re-runs SfS but does not redo the stereopsis. The following subsections now detail the modules used in this framework, other than the segmentation algorithm for which no further detail is needed, and the integration algorithm which is already fully detailed.

3.3.1 Stereopsis module

The entire DSI of a single pixel can not be reasonably represented using a single Gaussian. A single disparity value and its confidence can however, so a stereopsis algorithm is used to select a good disparity for each pixel, where good means close enough to ground truth to converge to it. Felzenszwalb & Huttenlocher[79] is used, a relatively simple stereopsis algorithm based on belief propagation. It is enhanced in various ways, most notably the use of Birchfield & Tomasi's[69] sampling invariant dissimilarity measure. The resulting disparities can therefore

[5]We calibrate the camera to get irradiance from pixel values.

be considered as ranges, ± 0.5 the given value, rather than as infinitesimal point matches. It is also extended to be hierarchical, to reduce runtime.

Stereopsis provides the mean for each pixel's Gaussian distribution, but not the standard deviation. Standard deviation is calculated by applying the Laplace approximation[133, p.213] to the DSI. The DSI is defined per-pixel using Luv difference, a Gaussian blur is then applied to the DSI, which acts to interpolate between pixels for non-integer disparities, and the second differential calculated for the mean point using central differencing. From the second differential standard deviation may be calculated by fitting a Gaussian with the same second differential, at the mean provided by stereopsis. Luv difference is taken to be a measure of negative log likelihood. If the standard deviation is too large, i.e. the pixel is in a smoothly shaded area, the standard deviation is set to ∞, indicating no information and allowing SfS to take over entirely. This is the same as occluded pixels, which are also assigned an evidence of $\psi_t(x_t, y_t) = \phi[0, 0]$.

3.3.2 Albedo estimation module

Under the Lambertian reflectance assumption SfS requires an albedo map as input. Colour is ignored, with irradiance taken to be a function, f, of the average channel strength

$$I = f\left(\frac{r+g+b}{3}\right) \tag{3.24}$$

This function, f, is calibrated offline by taking multiple photos of a white surface under constant lighting with only shutter speed changed on the camera, then fitting a polynomial (This is the camera response calculation method used in HDR photography, see Debevec[134] for instance.). We also assume that the geometric camera calibration is known; using this surface orientation may be calculated from a disparity map, $\hat{\mathbf{n}}(x, y)$. This is used not only by this module but also by the SfS module. For an arbitrary texture it is impossible to distinguish texture variation from shading; this is the basis of '$3D$' effects in user interfaces. It has been noted that in textured regions stereo matching is effective, so additional SfS information is only necessary in relatively uniform regions. Uniform regions allow us to ignore texture variation and calculate constant albedo values for segments.

We take the segmentation provided by mean shift[53] and use it to define a set of regions, R. Within each of these regions the colour and albedo are assumed to be uniform. The irradiance, I, however will vary across the region because of shading effects. In order to correctly compute the albedo of a region we need to account for shading effects using the Lambertian shading equation, 3.2. Accordingly, we can estimate the albedo at each pixel via the relation

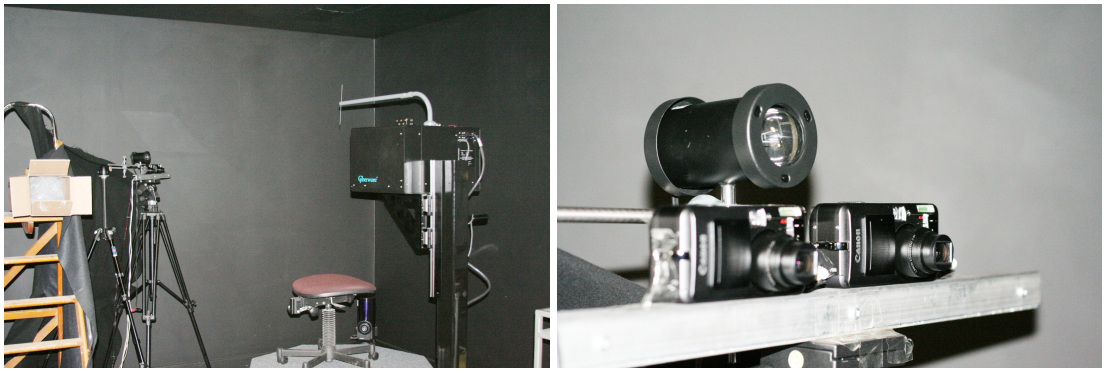
$$a(x,y) = \frac{I(x,y)}{\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}(x,y)} \tag{3.25}$$

where $\hat{\mathbf{l}}$ is the known direction to the light source. For individual pixels this is not reliable due to inaccurate normal estimation. As albedo is assumed constant an accurate estimate can be obtained by taking the median for an entire region - the median is used to obtain some robustness to the outliers. Due to rim lighting and noise mattering more as a pixel gets darker it is a weighted median, with each pixel weighted by the cosine of the angle between its surface orientation and the light source direction.

3.3.3 Shape-from-shading module

We adopt the Worthington & Hancock[21] iterative algorithm to solve for the field of surface normals by alternately smoothing and re-projecting onto the cone; specifically we use the DD9 smoothing variant[21]. This makes direct use of equation 3.2, which is a constraint on the angle between the surface normal and light source direction. Precisely stated, surface normals must lie on a cone whose angle is defined as $\cos^{-1}(I/a)$. Two modifications are made however, firstly no boundary constraint is used, secondly we initialise from the disparity map. A disparity map can be converted into a field of surface normals, as mentioned in the previous subsection. Worthington & Hancock consists of iterating between smoothing a field of surface normals and forcing them to obey the cone constraint, until it finally gets stuck in a local minima. By initialising with surface normals from the disparity map we hope to start closer to the global minima and hence ultimately get stuck closer to the global minima than otherwise. At the very

least this pushes the SfS result to take on the same concave/convex ambiguity assumptions as the stereopsis, for which that is not an ambiguity. Applying this method gives us fields of surface normals for either image, the framework only uses the left images orientation information however.



(a) Setup (b) Cameras

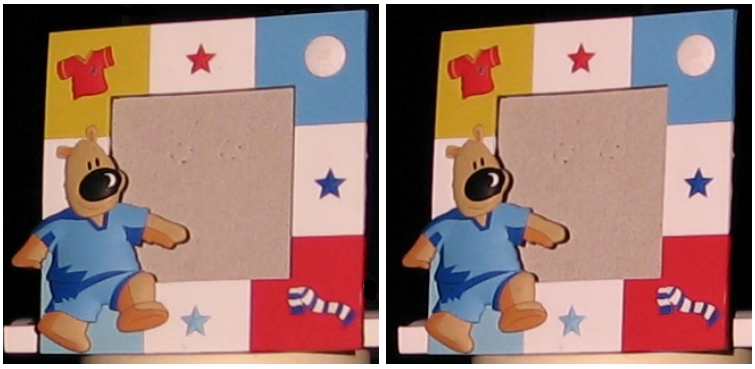
Figure 3.4: Setup used to capture the test data. The left image shows the entire setup - on the left are the cameras and light source, on the right the chair on which a human subject would sit and the 3D cylindrical laser scanner. The chair is replaced when non-humans are processed. In the right image a detail shot of the cameras is shown, with the light source just about close enough to be considered to be coming from the left camera.

3.4 Experiments

The documented algorithm is now applied to four real world inputs, one in each subsection of this section. First we have the frame input, which gives good results and provides ample opportunity to discuss the presentation. Following this a failed result is given, the plant pot, before we move to the watering can and head, which both work but include weaknesses worth discussing.

This test set was captured in a darkroom with a single light source. Two Canon PowerShot S70 cameras on a stereo bar captured the scene whilst a single light was positioned close enough to the left camera to be modelled as coming from the left camera, giving a light source direction of $[0, 0, 1]^T$. This provided input; to get ground truth disparity a Cyberware 3030 head scanner was used to capture a $3D$ model of the scene. Test objects were limited by the use of a head scanner to head sized objects, it is additionally a cylindrical scanner, so certain regions are occluded from the scanner and masked out from the ground truth disparity maps. Eyes were also removed from the head, as they cause issues due to the laser bouncing around inside them. The cameras were calibrated using multiple images of a football with markings on[6] - this allows the cameras to be calibrated

[6]The football is moved between each shot - this is necessary as a sphere is a degenerate shape for geometric calibration. It also improves stability by simply having a larger number of matches.



(a) Left (b) Right

Figure 3.5: Frame input images

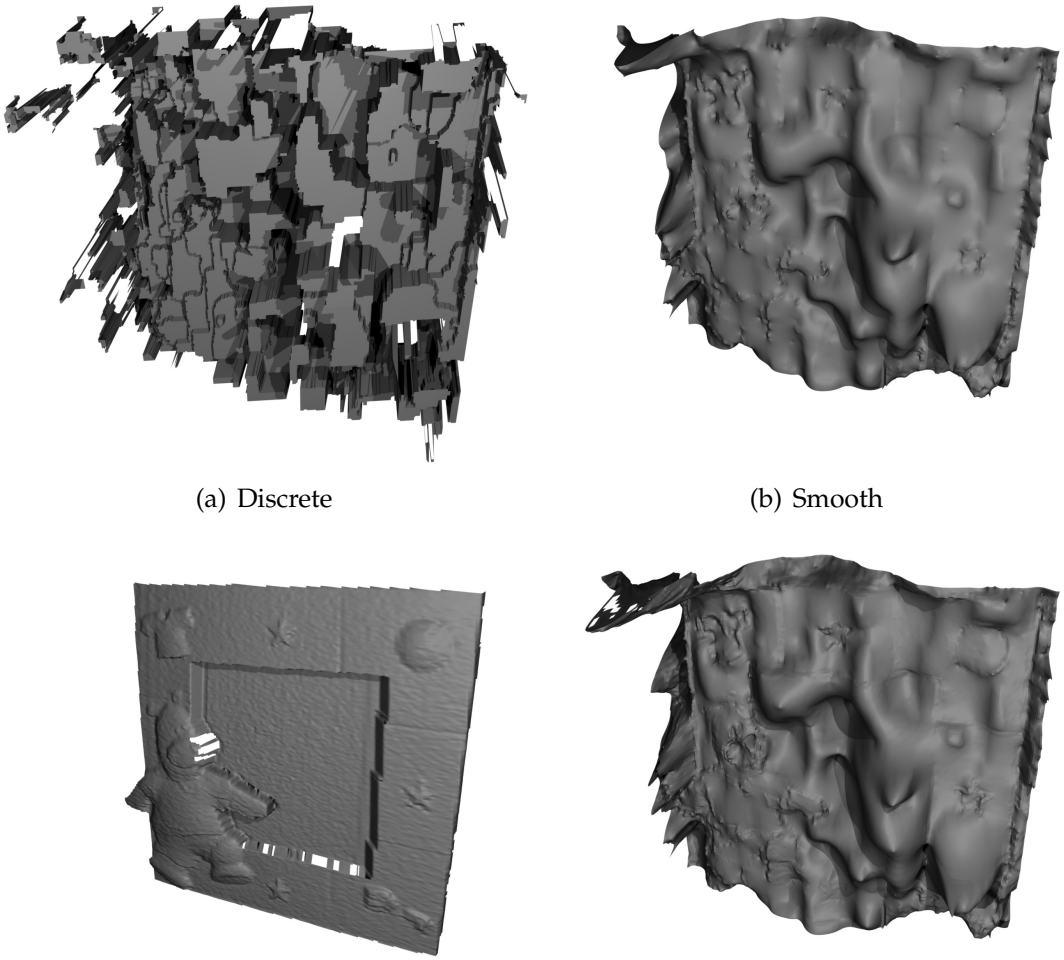
into the 3D scanners coordinate system for generating the ground truth disparity maps. Additionally the cameras are calibrated offline to determine the function from image value to irradiance. Pictures of the setup can be found in figure 3.4.

3.4.1 Frame input

The frame input is shown in figure 3.5 - the background, whilst mostly black anyway, has been masked out; the images rectified and the colours matched[7]. These inputs are run through the described algorithm, producing many outputs. Firstly stereopsis produces a discrete disparity map, referred to as *Discrete*, which is then smoothed to produce a smooth disparity map, *Smooth* and then, after albedo estimation and SfS, a disparity map integrating both stereopsis and SfS is produced, which is referred to as *Boot 1*. The iteration step, see figure 3.3(b), is then run six times, to produce *Iter 2-7*.

The final outputs, ignoring all the intermediate needle maps etc., are given in figures 3.6 and 3.9, as $3D$ renders of triangulated disparity maps. For completeness figure 3.7 shows some of the intermediate steps. Starting with figure 3.6 the first thing to note is that the discrete stereo map is not very aesthetically pleasing, unsurprising given its stepped nature, so improving on it means very little. It is the comparison of *Smooth* and *Boot 1* that is of the greatest interest - they are both calculated in the same way from the discrete disparity map, with the only

[7] A simple scaling so the average brightness for each colour channel is identical - this improves stereo matching slightly.

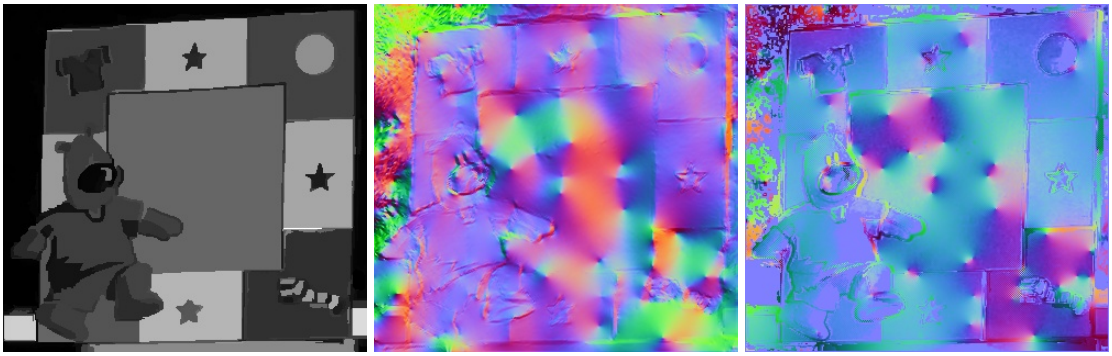


(a) Discrete (b) Smooth

(c) Ground Truth (d) Boot 1

Figure 3.6: Frame output renders, set 1. Note that to generate the ground truth image instead of using the 3D model directly the ground truth disparity map was triangulated, so it accurately represents the best that the algorithm could do.

differences between them the consequence of using SfS information for *Boot 1*. They are evidently very similar - the broad shape of the disparity input mostly remains, but focusing on the bear *Boot 1* shows a marginally better and more distinct shape. Larger improvement can be found if we move to analysing figure 3.9. Here the iterative process smooths the disparity map quite effectively, reducing the major mistakes of the stereopsis algorithm, whilst maintaining the detail provided by SfS - as a consequence the stars, ball and t-shirt all become visible. The bear itself forms a fairly reasonable shape, except at the nose where the dark colour proves unsuitable for SfS and noise takes over. It is certainly not perfect, and a long way from ground truth, but still a notable improvement over the original



(a) Albedo Map (b) Disparity Needle Map (c) SfS Needle Map

Figure 3.7: Frame intermediate samples. On the left we have the first albedo map estimated, in the centre the surface orientation directions calculated from the *Boot 1* disparity map, and on the right the surface orientation directions for the SfS result that goes into *Boot 1*. Needle maps are represented with red mapping to the x component, green the y component and blue the z component of the surface normal. Red and green map $[0, 225] \rightarrow [-1, 1]$, whilst blue, seeing as normals can not face away from the camera, maps $[0, 255] \rightarrow [0, 1]$.

discrete or smoothed disparity maps. Flaws are visible, but we save discussion for the other inputs.

Some discussion of the intermediate results given in figure 3.7 is now had. No ground truth is available for the albedo map, but qualitatively it looks entirely plausible. Whilst recalculated for further iterations it shows little change from the initial estimate given. The middle needle map is calculated from the disparity map associated with 3.6(d), and gives another view of the detail extracted, especially at the edges of entities such as the star, bear etc. SfS takes a disparity needle map[8] and iterates until convergence to the right hand needle map. Large global changes are evident between the maps, whilst fine detail appears to remain. This is as intended - the disparity information provides global detail overriding that provided by SfS, which has evidently assumed the plane to be closer to head on to the camera rather than at its actual angle, whilst the fine detail from SfS is used to interpolate the shaded areas correctly and with significantly more detail.

Quantitative results are given in figure 3.8 - inlier percentages are used, for var-

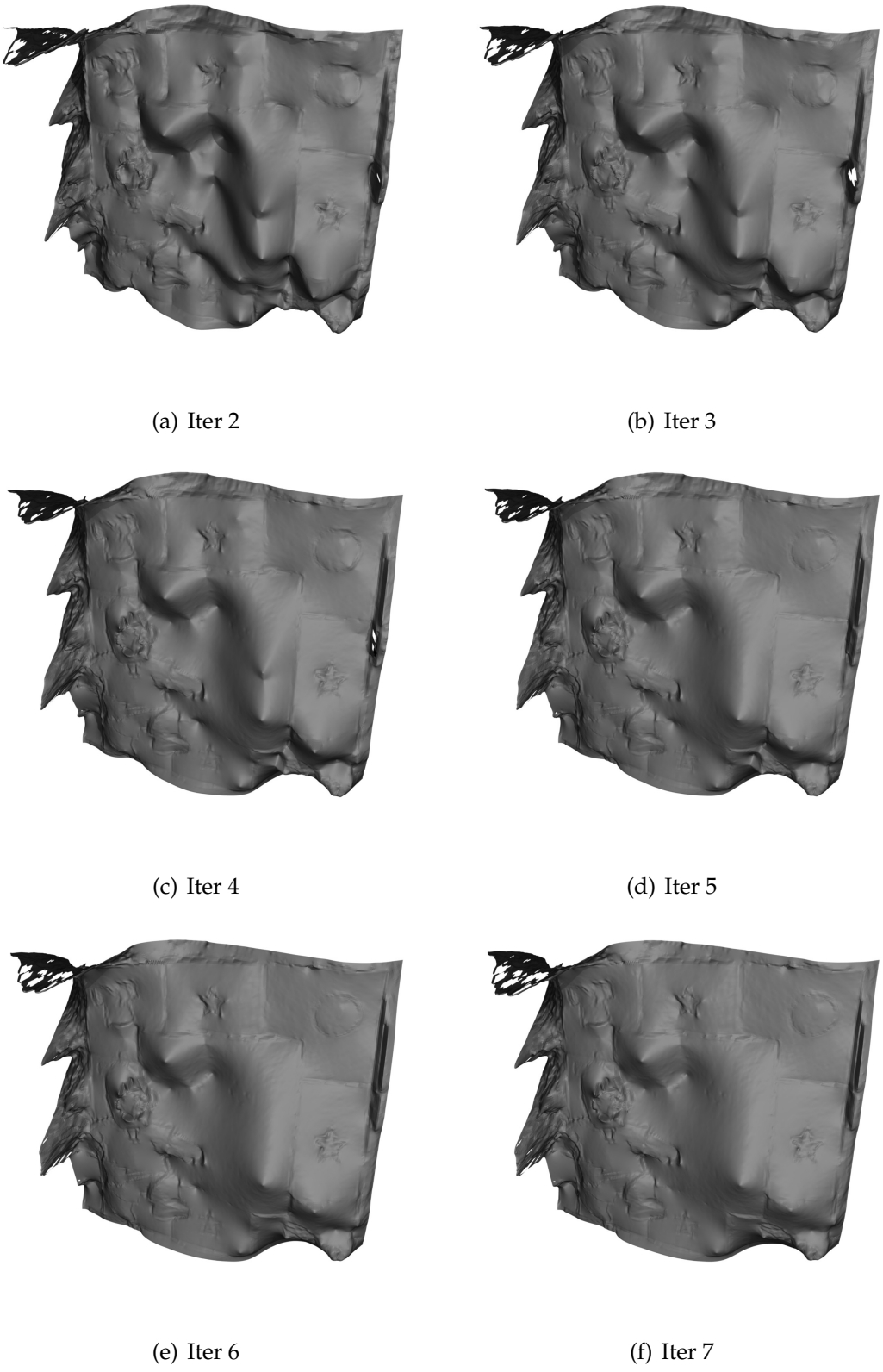
[8]In the case of the one shown the disparity needle map associated with the smoothed disparity map.

Frame	< 0.125	< 0.25	< 0.5	< 1	< 2	< 4
Discrete	7.1	13.9	26.6	45.3	62.1	77.2
Smooth	9.3	19.1	35.5	59.3	75.4	88.3
Boot 1	10.2	19.3	35.1	57.9	73.9	86.7
Iter 2	9.9	19.1	36.0	61.8	76.7	89.2
Iter 3	10.1	19.4	36.9	62.6	77.3	90.0
Iter 4	10.3	19.8	37.0	63.7	78.1	90.8
Iter 5	10.2	19.7	36.8	63.8	78.6	91.6
Iter 6	10.0	19.2	36.7	63.8	79.0	92.2
Iter 7	9.9	18.7	36.9	63.7	79.5	92.7

Figure 3.8: Quantitative results for the frame input. Each row shows the results for a specific disparity map. Each column shows a percentage of inlier pixels for a definition given at the top of each column, in terms of absolute disparity difference. For instance, the first column gives the percentage of pixels where the disparity value is less than 0.125 from ground truth. Only pixels where ground truth is defined are included. The bars behind the number indicate the percentage graphically, with the highest percentage(s) in each column green rather than blue. First row is the discrete disparity result, direct from the stereopsis algorithm. Second is the smoothed result, marked as *Output without SfS* on figure 3.3(a) and third, *Boot 1*, is from the same figure, where it is marked as *Output with SfS*. Following that are *Iter 2-7*, these being applications of the iterative step from figure 3.3(b).

ious definitions of inlier[9]. They clearly show that the algorithm is an improvement on either discrete or smoothing, though unfortunately the improvement peaks before dropping off again, presumably due to over smoothing. This problem is discussed further in the following subsections. Comparing *Smooth* to *Boot 1* shows that for fine detail (Inlier definitions of < 0.125 and < 0.25.) the use of SfS has caused a slight improvement. For the larger inlier definitions however smoothing is, initially at least, better. This is probably because the smoothing strength is variable, and at its highest when the normal faces the camera - this means that the smoothing without SfS is actually a stronger operation, and can hence reduce some of the large errors made by the stereopsis algorithm more effectively.

[9]The Middlebury stereo test gives outlier percentages, by default identical to the < 1 column of figure 3.8, making the results comparable after converting from inliers to outliers. However, the focus here is on showing improvement from using SfS - it does not matter that an older stereopsis algorithm has been used even though as a consequence the scores given do not even make the current chart.



(a) Iter 2

(b) Iter 3

(c) Iter 4

(d) Iter 5

(e) Iter 6

(f) Iter 7

Figure 3.9: Frame output renders, set 2.

(a) Left (b) Right

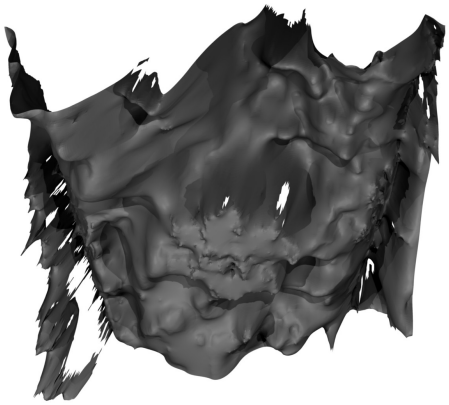
Figure 3.10: Plant pot input images

3.4.2 Plant pot input

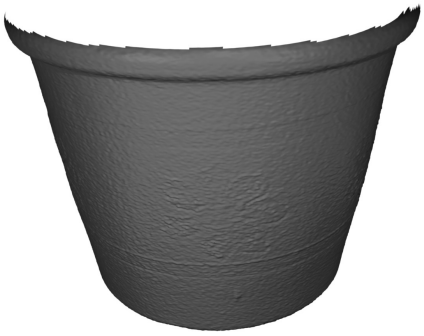
We present this identically to the frame - inputs are in figure 3.10, $3D$ renders of outputs in figures 3.11 and 3.13, intermediate outputs in 3.12 and, finally, quantitative results in figure 3.14. The plant pot is given as an example of failure - it can be made to work by switching to a different stereopsis algorithm. This fact points towards the issue - the stereopsis result is bad, consequentially the SfS is initialised badly and fails to converge to anything useful. As a result it never corrects for the initial stereopsis errors - if the algorithm initialises too badly neither algorithm will be able to correct for the error due to the interdependency. In the frame image there were bad stereo results however, and they were corrected to some degree with good SfS results, so there is some corrective capability[10]. Stereopsis does produce a good result for the flower on the plant pot, where examination of the SfS results (Figure 3.12(c).) shows that SfS mostly fails for the flower, yet the flower survives the integration process - this is an example of SfS failing when stereopsis gives good results, and the consequence is a reasonable overall result. In conclusion the algorithm can not cope with a double failure, and one algorithm (stereopsis) failing can push the other (SfS) to fail - but if only one algorithm fails then the other can correct.

[10]Note also that the bottom right corner of the frame never corrected itself, so you can have total failure as well as one algorithm correcting the other in the same input.

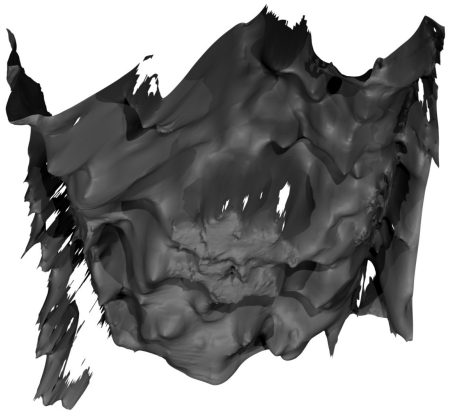
(a) Discrete



(b) Smooth



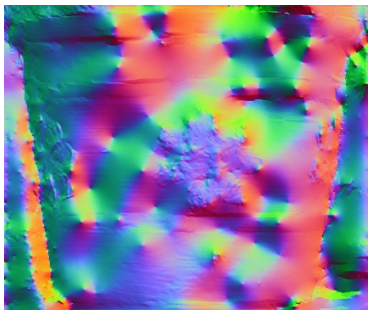
(c) Ground Truth



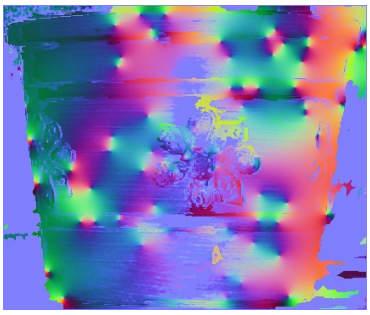
(d) Boot 1

Figure 3.11: Plant pot output renders, set 1.

(a) Albedo Map

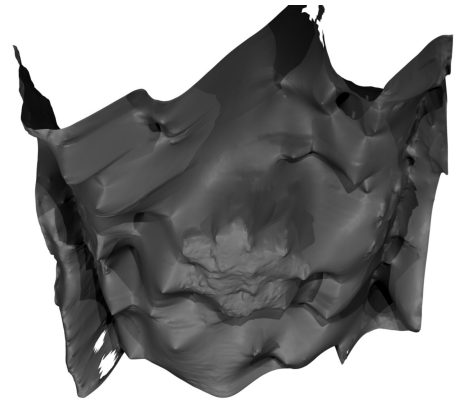


(b) Disparity Needle Map

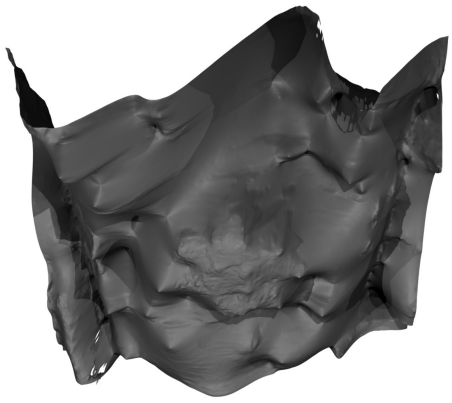


(c) SfS Needle Map

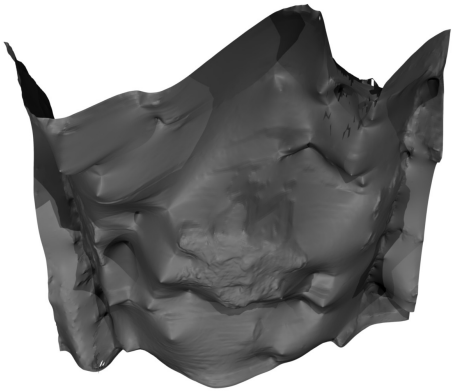
Figure 3.12: Plant pot intermediate samples.



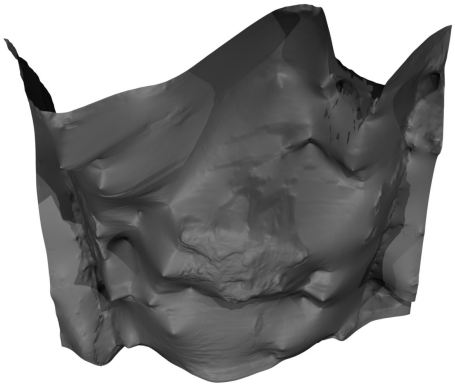
(a) Iter 2



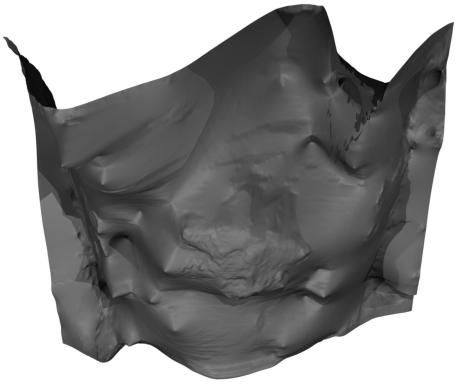
(b) Iter 3



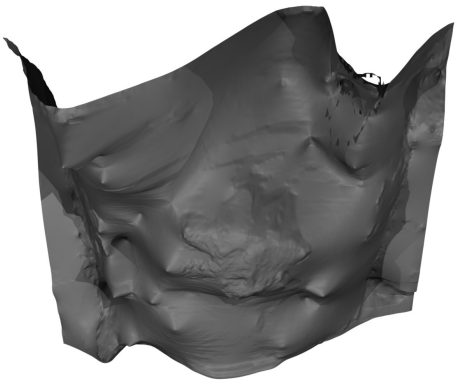
(c) Iter 4



(d) Iter 5



(e) Iter 6



(f) Iter 7

Figure 3.13: Plant pot output renders, set 2.

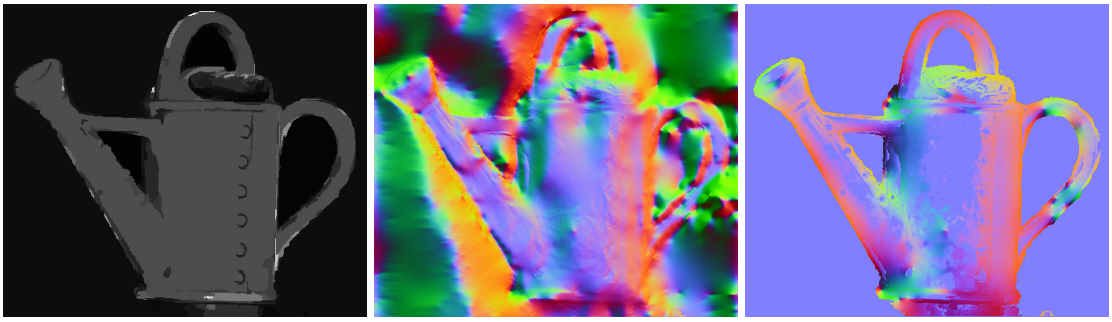
Plant pot	< 0.125	< 0.25	< 0.5	< 1	< 2	< 4
Discrete	4.0	7.9	15.0	24.7	38.4	55.9
Smooth	4.7	9.4	17.2	29.1	46.6	64.8
Boot 1	4.3	8.7	16.7	29.8	47.6	64.4
Iter 2	4.3	8.6	15.8	27.6	46.5	66.2
Iter 3	4.0	7.7	14.9	26.4	44.8	66.8
Iter 4	3.8	7.4	14.2	25.3	44.7	67.1
Iter 5	3.5	7.1	13.6	24.9	44.1	67.8
Iter 6	3.5	7.0	13.2	24.8	43.7	68.8
Iter 7	3.5	6.9	13.1	25.0	43.6	70.0

Figure 3.14: Quantitative results for the plant pot input. See figure 3.8 for explanation.

(a) Left (b) Right

Figure 3.15: Watering can input images

Examining the actual input in figure 3.10 there is a specularity, which has been picked up to some degree by the albedo map in figure 3.12(a). This is problematic as both algorithms assume no specularities, and this probably contributed to the overall failure. Quantitative results are given in figure 3.14. It is not that surprising that simple smoothing did better most of the time. The improvement shown at the higher outlier thresholds for the actual algorithm can probably be put down to it simply smoothing the input and damping away some stereopsis errors.



(a) Albedo Map (b) Disparity Needle Map (c) SfS Needle Map

Figure 3.16: Watering can intermediate samples.

3.4.3 Watering can input

The watering can is presented as before, in figures 3.15, 3.17, 3.18, 3.16 and 3.19. This input is very SfS friendly, as its mostly without texture, though it is quite dirty[11]. It also has sharp specularities; they do not appear to be much of a problem however. Initial results, *Boot 1*, do not look much better than the smoothed results, though the quantitative results indicate an across the board improvement with the inclusion of SfS, albeit a fairly minor one for larger inlier thresholds. However, as iterations continue it smooths out and ends up at a fairly reasonable shape, though a crease forms at the line of dimples, and the base takes on a spherical rather than cylindrical shape. Dimples are never extracted unfortunately, though a deformation is visible at each. The handles, whilst showing some distortion, appear correct if rather flat, and the spout has a cylindrical shape. Handle flatness is likely due to their size - the SfS information will not overcome the stereopsis as the nearby edges indicate that the stereopsis is reliable.

Quantitative results indicate that the best answer was in one of the earlier iterations, rather than the last which could be argued to be the qualitative best, though there is not much in it. This is explained by the smoothing moving good disparity value away from the correct answer when they are near to bad disparity values, dropping the inlier percentage; i.e. this is a consequence of the algorithms assumption that the scene is smooth (No occlusion.) with no bad disparity values.

[11] All parameters are the same for all inputs presented, except for one. This is a multiplicative term of the standard deviation assigned to disparity values, which effectively controls the relative confidence of the SfS and stereopsis. Frame and pot have the same value, this input and the next, head, have a smaller value as they have more texture, allowing the stereopsis to do a better job.

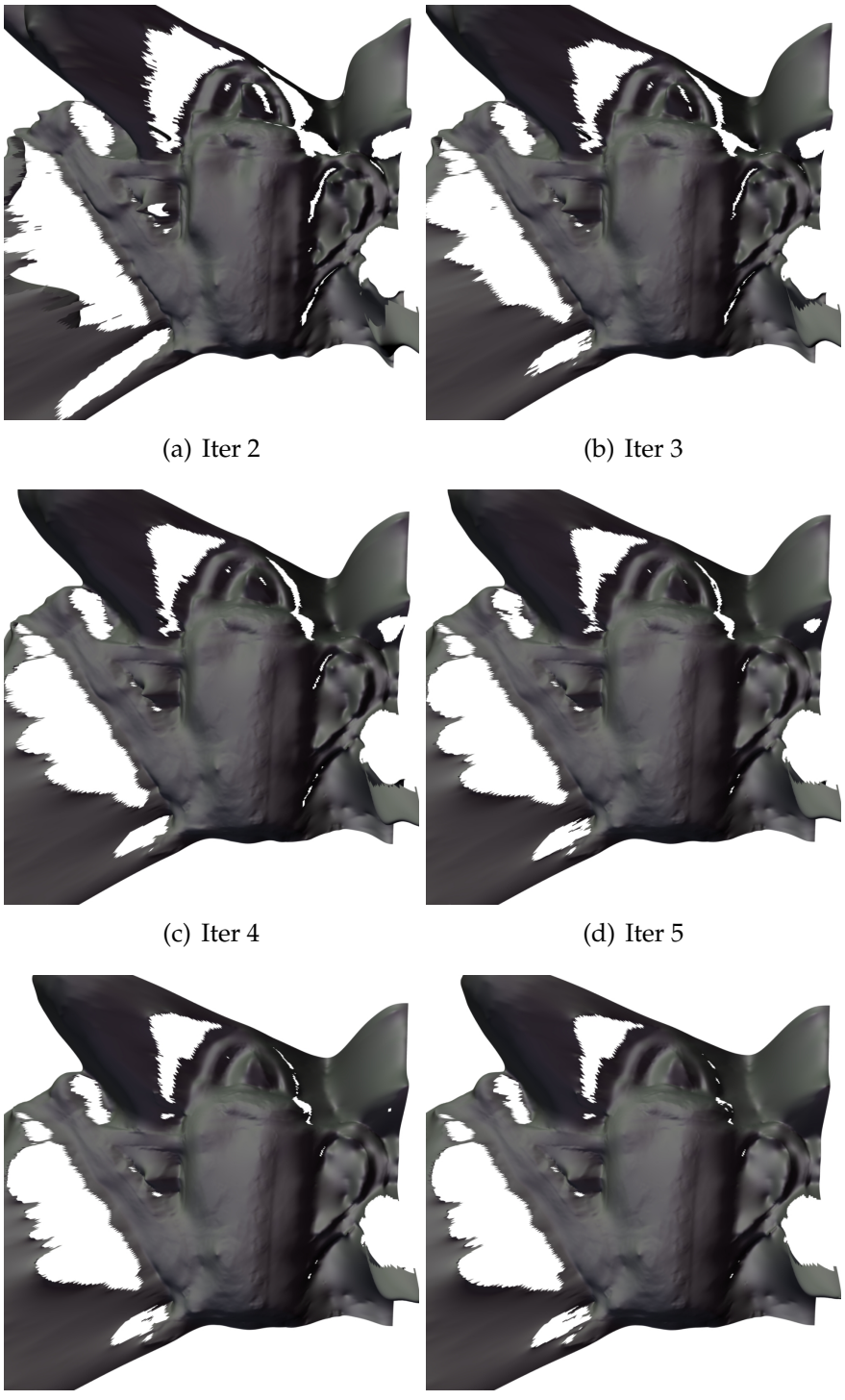
(a) Discrete

(b) Smooth

(c) Ground Truth

(d) Boot 1

Figure 3.17: Watering can output renders, set 1.



(a) Iter 2 (b) Iter 3

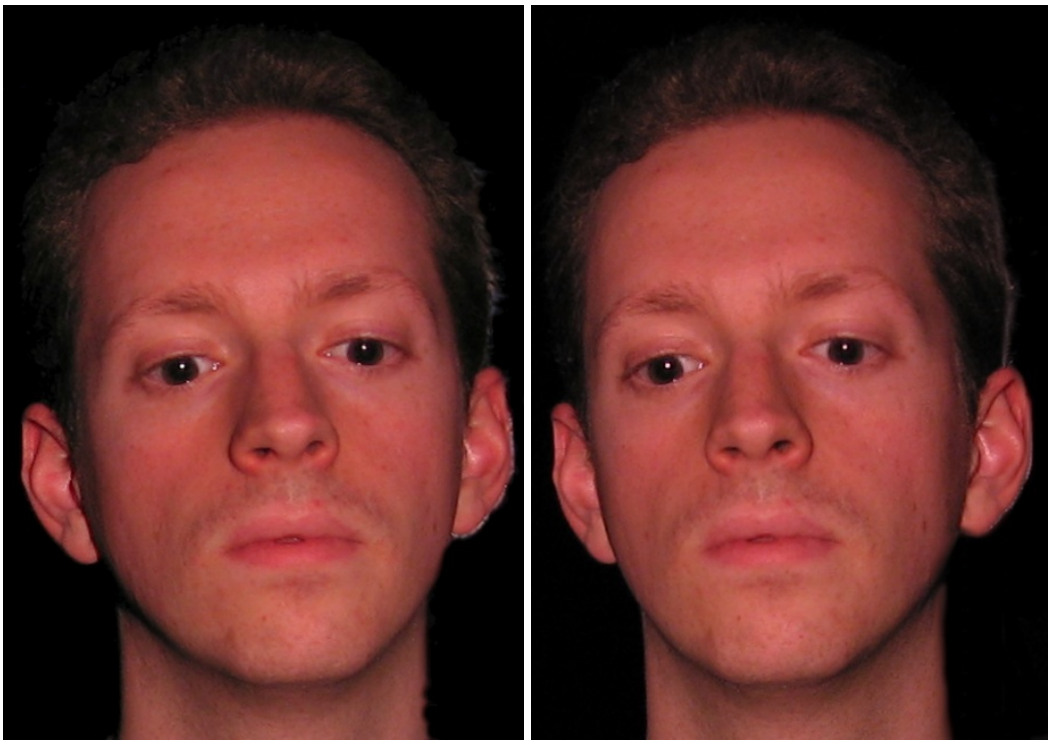
(c) Iter 4 (d) Iter 5

(e) Iter 6 (f) Iter 7

Figure 3.18: Watering can output renders, set 2.

Watering can	< 0.125	< 0.25	< 0.5	< 1	< 2	< 4
Discrete	8.3	16.2	29.8	52.3	76.7	92.2
Smooth	6.6	14.3	28.7	55.5	78.0	93.7
Boot 1	8.6	16.2	30.3	56.0	78.6	93.8
Iter 2	8.6	16.6	29.8	55.4	78.7	93.2
Iter 3	8.6	16.8	30.7	54.2	78.6	92.5
Iter 4	8.3	16.1	31.1	52.7	77.2	91.7
Iter 5	8.3	16.1	31.5	52.1	75.9	91.1
Iter 6	8.1	15.9	30.8	51.9	74.7	90.6
Iter 7	7.7	15.7	30.2	51.1	73.6	89.9

Figure 3.19: Quantitative results for the watering can input. See figure 3.8 for explanation.



(a) Left (b) Right

Figure 3.20: Head input images

(a) Front (b) Side

Figure 3.21: Final iteration of head image rendered with the original images as textures.

3.4.4 Head input

The same presentation is used as before, in figures 3.20, 3.23, 3.25, 3.27 and 3.28. Additionally side renders are provided to improve perception of shape, in figures 3.24 and 3.26. This head test is important as analysing heads is not only of value but an area where combining stereopsis and SfS has great potential - skin is a mostly uniform material, so shading is detectable, making SfS workable, but also often has enough blemishes and 'features' for stereopsis to produce good results. Of course skin is an example of a surface with sub-surface scattering (See section B.1.), which causes problems for SfS, and oily skin can exhibit strong specularities, which are problematic for both. The results do not provide evidence of either being a significant problem however.

The results are quite similar in quality and failures as the watering can. Firstly the qualitative difference between the smoothed and SfS smoothed outputs is small, though quantitative results indicate the SfS result to be better. Iteration leads to a smooth result with most details intact, though the right side (As for the image, left for subject.) eye socket suffers somewhat. Again, quantitatively the

	Stereopsis	SfS	Integration
Frame (331X317) [1296]	10s	373s	34s
Plant pot (428X359) [1574]	14s	607s	60s
Watering can (464X399) [1726]	19s	372s	78s
Head (459X647) [2212]	53s	2569s	319s

Figure 3.22: Timing results for each of the inputs, per use of a module. Each input has its left image resolution next to it in the form (width X height), and its iteration count, in the form [iterations]. SfS and integration are taken as medians over all runs. All times given in seconds.

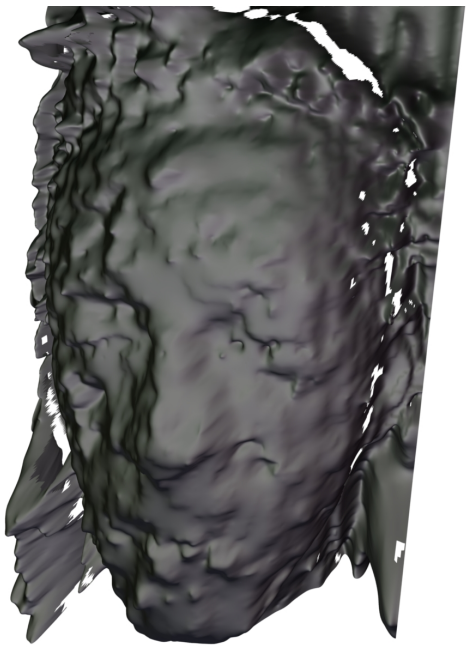
results suffer with further iterations despite qualitative improvement. Also note that the disparity needle map and SfS needle map are very similar, indicating that the two algorithms are producing similar answers. Ultimately it produces a 3D shape that is instantly recognisable as a head and arguably recognisable as the specific head in question. If the final result is textured with the original image, as in figure 3.21, the result is quite convincing, as long as you avoid looking at the ears.

3.4.5 Resource Usage

Figure 3.22 gives the relevant timing information[12]. This approach does consume a lot of time compared to the stereopsis algorithm. However, they all involve passing messages between adjacent nodes and the stereopsis algorithm is the only one which uses a hierarchical structure, conferring it a massive advantage; it has also been heavily optimised. Additionally, the iteration counts, which are for the SfS and Integration algorithms, have been set much higher than need be as no convergence detection has been used. There is certainly no reason why the integration step could not be optimised to a time comparable to the stereopsis algorithm, and a much faster SfS algorithm is discussed in the next chapter. The message passing nature of these algorithms makes them easy to adapt to parallel systems, which is advantageous. Resource usage of the integration algorithm is relatively small - it only requires 14 floats per pixel at runtime, which is less than the stereopsis algorithm.

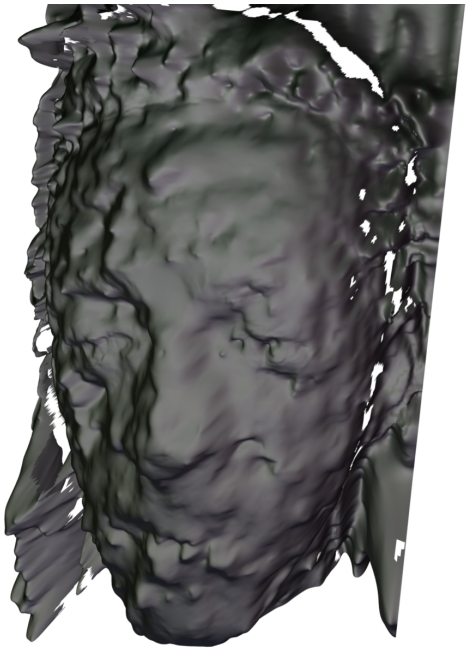
[12]Run using a single core of a Core 2 Duo 2Ghz with 1Gb of RAM.

(a) Discrete



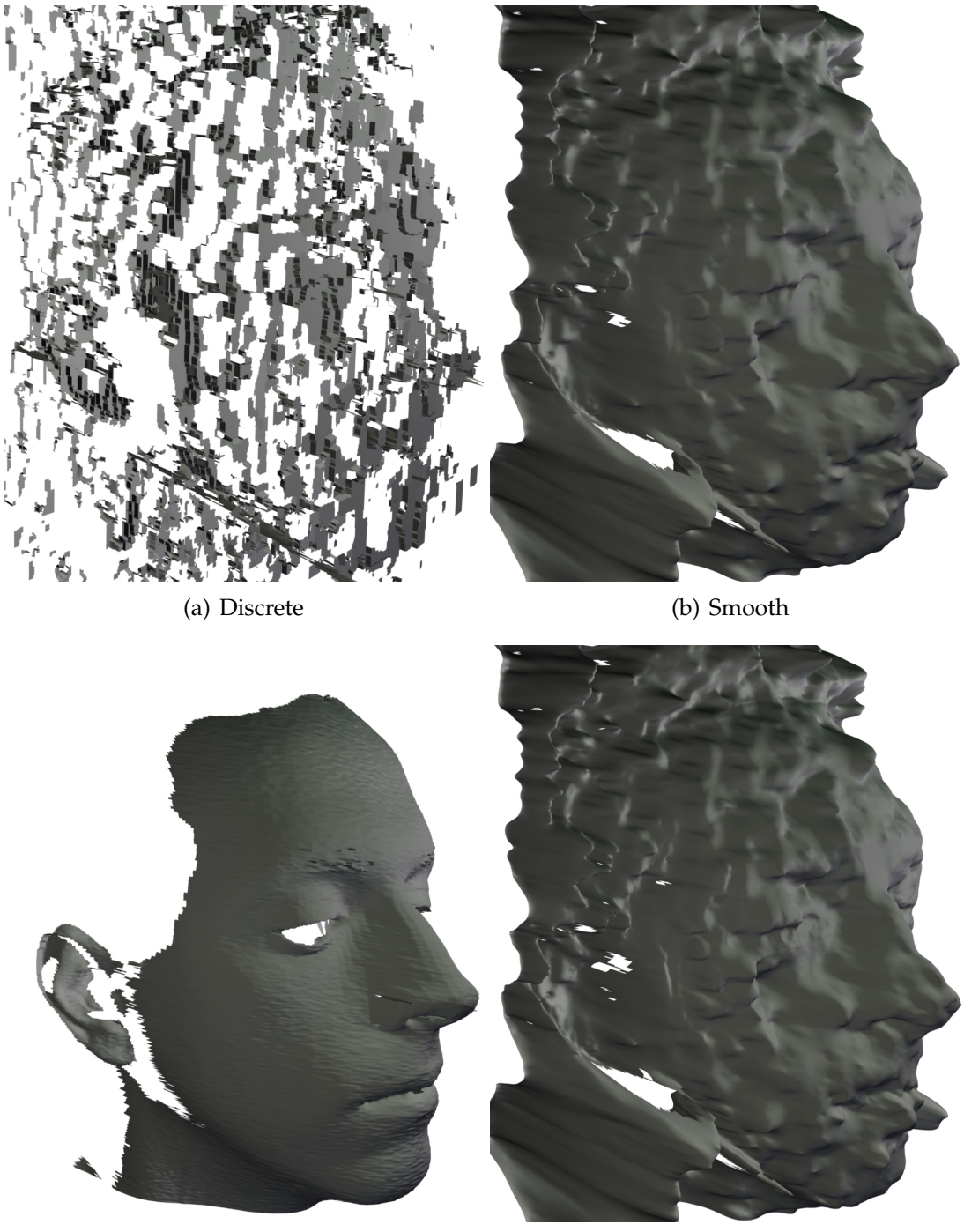
(b) Smooth

(c) Ground Truth



(d) Boot 1

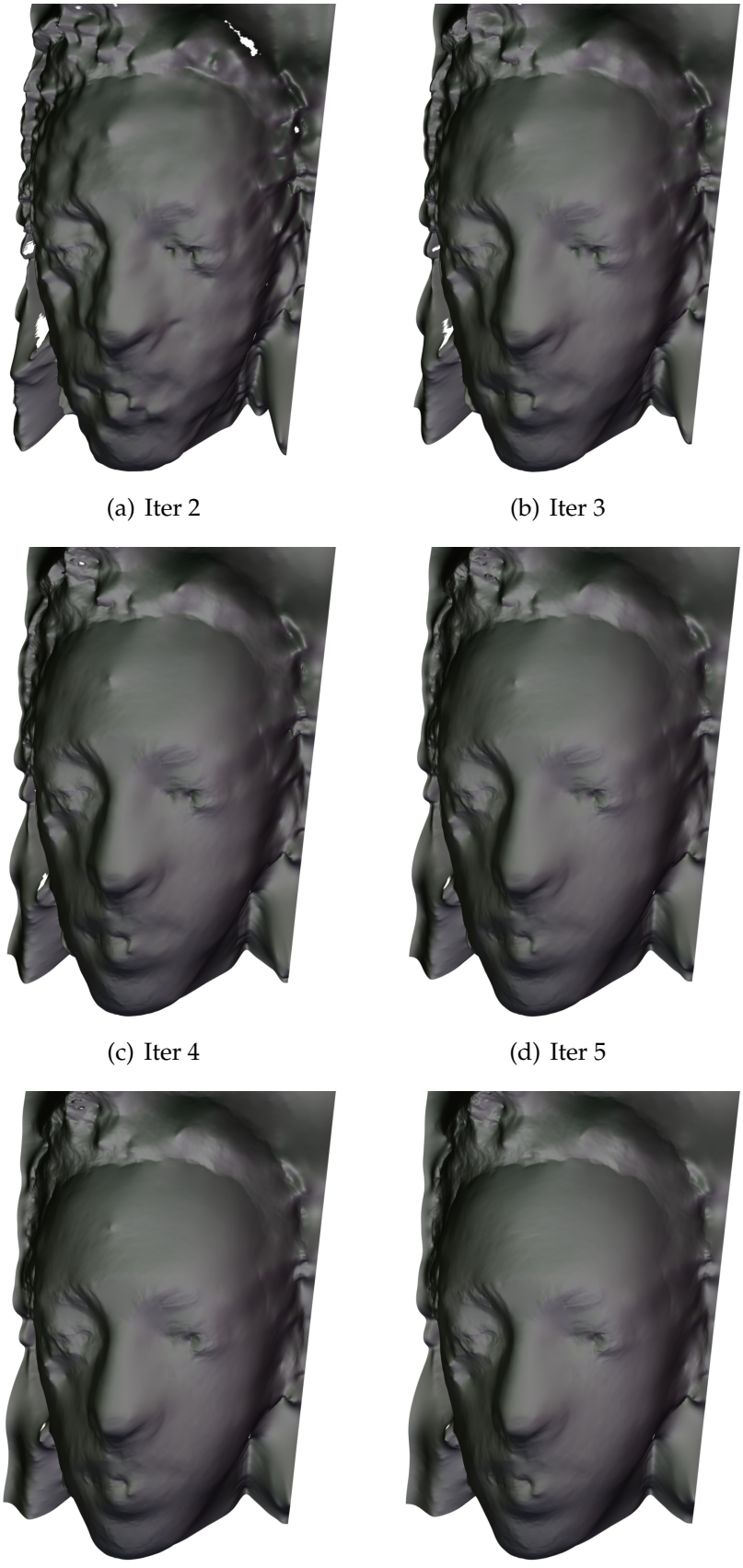
Figure 3.23: Head output renders, set 1.



(a) Discrete (b) Smooth

(c) Ground Truth (d) Boot 1

Figure 3.24: Head output renders, set 1, view from the side.



(a) Iter 2 (b) Iter 3

(c) Iter 4 (d) Iter 5

(e) Iter 6 (f) Iter 7

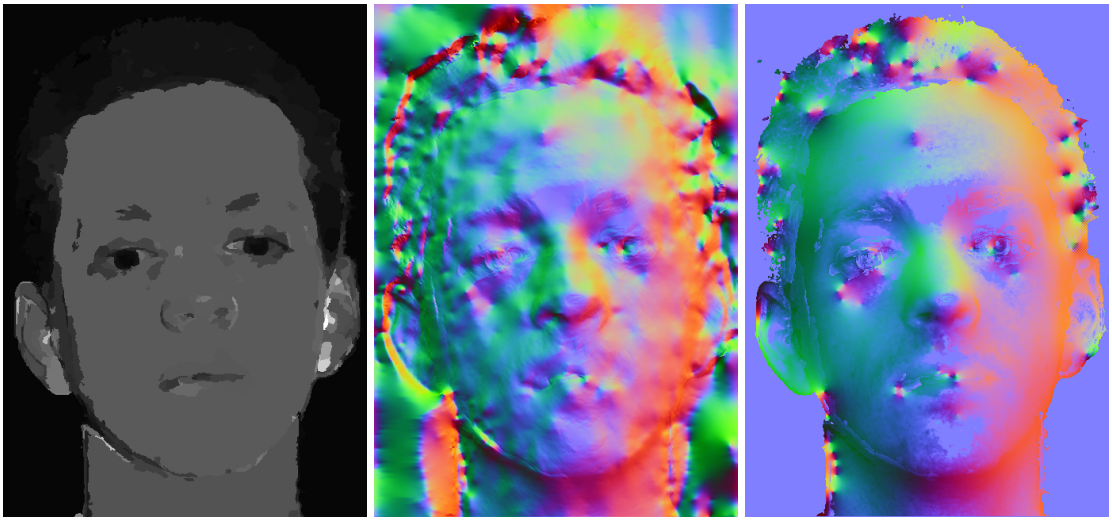
Figure 3.25: Head output renders, set 2.

(a) Iter 2 (b) Iter 3

(c) Iter 4 (d) Iter 5

(e) Iter 6 (f) Iter 7

Figure 3.26: Head output renders, set 2, view from the side.



(a) Albedo Map (b) Disparity Needle Map (c) SfS Needle Map

Figure 3.27: Head intermediate samples.

Head	< 0.125	< 0.25	< 0.5	< 1	< 2	< 4
Discrete	4.4	8.7	17.5	33.2	57.5	84.7
Smooth	5.1	10.4	21.2	41.5	72.7	95.6
Boot 1	5.7	11.3	22.6	43.8	74.0	95.7
Iter 2	6.5	12.6	23.8	45.0	76.4	96.4
Iter 3	6.0	12.2	23.9	45.2	76.3	96.3
Iter 4	5.8	11.9	23.7	45.5	75.8	96.0
Iter 5	5.7	11.5	23.5	45.0	75.2	95.6
Iter 6	5.8	11.4	23.3	44.4	74.5	95.2
Iter 7	5.9	11.6	22.9	43.9	73.9	94.7

Figure 3.28: Quantitative results for the head input. See figure 3.8 for explanation.

3.5 Conclusions

This chapter has presented a method for integrating shape-from-shading information with stereo information. It runs the algorithms separately before performing the merge using Gaussian belief propagation. The method efficiently delivers a continuous estimate of disparity and is relatively easy to implement. Our results show an improvement in the fine surface details when shading information is used, leading to more visually pleasing models. A quantitative weakness appears, as whilst the algorithm shows initial improvement over simple smoothing further iterations produce an aesthetically preferable but numerically inferior result. Both algorithms contribute to the output, with evidence showing regions where one algorithms failure is masked by the others success. There are a number of issues that we will now iterate however:

- The stereopsis algorithm is probably not appropriate - it guesses disparities where it should give up and let SfS take over. A custom stereopsis algorithm designed for the task at hand could confer improvement by actually being simpler and making no effort to guess. This could be a local steropsis algorithm that uses cost aggregation only, and only keeps matches that it is sufficiently confident in.

- The shape-from-shading algorithm does not provide confidence information - setting all values of the needle map to the same confidence makes selecting a confidence parameter problematic as it will be too strong in some areas and too weak in others. Fixing this would be likely to reduce the problem where results get worse with too many iterations.

- The SfS is initialised from the current disparity result, which can lead to both algorithms failing - a SfS algorithm that avoids this would be advantageous.

- Specularities are a problem. Adding a new module to the framework would probably work best - a module to remove them could be introduced, and could make use of the (unreliable) disparity map as a source of surface orientation information.

- Double failure currently results in garbage. If both algorithms are giving confidence estimates then a threshold can be used and simple interpolation used when this happens.
- Light source direction is currently an input. Having the system infer this would be ideal.
- No correlation is assumed between dz/dx and dz/dy for a given pixel, which is not correct. This could be fixed with GBP using a clique size of three, which would unfortunately cause a large slow down in the algorithm.

Much further work has been implied above. The weaknesses we will highlight are firstly the requirement that light source direction be provided - this is not ideal as this information is often not available. Also both the stereopsis and SfS modules show weaknesses that can be improved upon. With regards to stereopsis in the current approach it is not involved beyond the initial bootstrap step - a stereopsis algorithm that takes expected disparity differences as an input and produces a continuous disparity map as output could replace the integration module however. SfS is one of the greatest weaknesses however, due to it lack of confidence information and dependency on stereopsis, and will be the subject of the following chapter.

Chapter 4
Shape-from-Shading

WEAKNESSES were observed with the SfS module in the previous chapter; consequently in this chapter a new SfS algorithm is proposed. The observation was made in the conclusion of the previous chapter, section 3.5, that SfS is initialised using the disparity information, with the consequence that a bad disparity result can also break the SfS algorithm. We want to avoid this, however, we don't want to throw away the disparity information entirely as it can improve the SfS result[1]. The proposed solution is to have a probabilistic algorithm which uses the disparity information, but with a measure of confidence, to avoid bad disparity values causing SfS failure. Lending further weight to the advantages of a probabilistic algorithm is the potential to output a probabilistic confidence measure with the results, to be fed into the combining step of the SfS & stereopsis algorithm. The presented algorithm is therefore probabilistic, and like the previous chapter constructs a pairwise Markov random field over the image pixels, which is solved with belief propagation. SfS is concerned with surface orientation however, which motivates the need for a probabilistic surface orientation representation. Directional statistics are therefore introduced, and, consequentially, a large part of this chapter is concerned with using directional statistics within a belief propagation framework.

Despite this algorithm being designed with consideration towards future integration with the previous chapters SfS and stereopsis combining algorithm the presentation here is kept separate, focusing on it as an independent SfS algorithm. The next section formulates the algorithm, or more accurately references back to

[1] Additionally the iterative refinement only makes sense because the SfS result uses the stereopsis information; a non-iterative version is easy to define however.

and restates the assumptions and ideas given in previous chapters that are immediately relevant. Following, in section 4.2, the algorithm itself is presented, though the following section 4.3 details how to pass messages consisting of directional distributions. This is itself a novel contribution, which has the potential to be applied to other problems. Finally, section 4.4 gives experimental results and it is concluded in section 4.5.

4.1 Specifics

An algorithm to solve the SfS problem under Horns original assumptions[6], as given in section 2.1, is proposed. In consequence the Lambertian shading equation applies, which is given as[2]

$$I_{x,y} = a\hat{\mathbf{l}} \cdot \hat{\mathbf{n}}_{x,y} \tag{4.1}$$

where $I_{x,y}$ is the irradiance provided by the input image, a is the albedo and $\hat{\mathbf{l}} \in \mathbb{R}^3, |\hat{\mathbf{l}}| = 1$ is the direction to the infinitely distant light source; these are all inputs. This variant of the shading equation fixes the light source strength to 1, as when considering an image there is no method to determine the ratio between light source strength and albedo. $\hat{\mathbf{n}}_{x,y} \in \mathbb{R}^3, |\hat{\mathbf{n}}_{x,y}| = 1$ is the normal map to be inferred as the algorithm's output. The normal map can be integrated as a further step to obtain a depth map - this problem is deemed separate to this work, but the algorithm used for presenting results later in section 4.4 is outlined in subsection 3.2.1. Substituting the dot product with the cosine of the angle between the two vectors you get

$$\frac{I_{x,y}}{a} = \cos\theta_{x,y} \tag{4.2}$$

where θ is therefore the angle of a cone around $\hat{\mathbf{l}}$ which the normal $\hat{\mathbf{n}}_{x,y}$ is constrained to - this is the cone constraint[21], previously given in sub-subsection 2.1.2.2. It leaves one degree of freedom per pixel that is not constrained by the available information; the presented algorithm takes the typical approach (Subsection 2.1.2.) and uses smoothing as a further source of information.

Key to the presented approach is the use of a directional distribution, appendix D, which allows the representation of surface orientation with a single random variable, rather than the two random variables, $\delta x/\delta z$ and $\delta y/\delta z$, that are used in Potetz[26] and many others. Specifically, the eight parameter Fisher-Bingham distribution [FB_8] is used, as documented in section D.2. This choice is made for two reasons - firstly, when multiplied with itself, you get another FB_8distribution; and secondly, using the sub-model of Bingham-Mardia distributions the cone constraint may be represented. Representing the cone constraint is of course

[2]See section B.2 for more details.

needed to provide the prior on each pixel, whilst the multiplication property is needed for belief propagation.

Belief propagation [BP], appendix C, is used to solve a Markov random field [MRF], section C.1, that represents the problem at hand. Each pixel has a prior expressing the cone constraint, and the compatibility between adjacent pixels provides a smoothing term. An additional post-processing step, given in subsection 4.2.3 and also using belief propagation, is required, to intelligently select output surface orientations from the posterior distributions.

In appendix C, where belief propagation is detailed, factor graphs are used; here pairwise MRF are used instead, and so the core equations are adjusted accordingly. This was previously done in subsection 3.2.2, but is repeated here in part to maintain independent presentation but also to introduce the equations in directional terms. Ultimately, a pairwise MRF represents

$$P(\mathbf{x}) = \prod_{v \in V} \psi_v(\mathbf{y}_v) \tag{4.3}$$

where $\mathbf{x}$ is a set of random variables and $\forall v; \mathbf{y}_v \subset \mathbf{x} \wedge \texttt{card}(\mathbf{y}_v) \leq 2$. This is equation C.1 with the additional requirement that the ψ functions can not involve more than two variables, hence the use of the term *pairwise*. The belief propagation equations find a solution that, approximately, maximises $P(\mathbf{x})$ marginalised over each variable. From node p to node q at iteration t the message passed is[77]

$$m^t_{p \to q}(\hat{\mathbf{x}}_q) = \int_{\hat{\mathbf{x}}_p} \psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q) \psi_p(\hat{\mathbf{x}}_p) \prod_{u \in (N-q)} m^{(t-1)}_{u \to p}(\hat{\mathbf{x}}_p) d\hat{\mathbf{x}}_p \tag{4.4}$$

where $\psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q)$ is the compatibility between adjacent nodes, $\psi_p(\hat{\mathbf{x}}_p)$ is the prior on each node's orientation and N is the 4-way neighbourhood of each node. Once message passing has iterated sufficiently for convergence to occur the belief at each node is

$$b_p(\hat{\mathbf{x}}_p) = \psi_p(\hat{\mathbf{x}}_p) \prod_{u \in N} m_{u \to p}(\hat{\mathbf{x}}_p) \tag{4.5}$$

4.2 Method

We construct a graphical model, specifically a pairwise Markov random field on a grid, that is equivalent to figure 3.1. Each node of the model is a random variable that represents the unknown surface orientation of a pixel. Continuous sum-product belief propagation, as described above and later in section C.3, is then used to determine the marginal distribution for each node, a FB_8 distribution. There are two equations from equation 4.4 yet to be defined, $\psi_p(\hat{\mathbf{x}}_p)$ and $\psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q)$. The prior on surface orientation, $\psi_p(\hat{\mathbf{x}}_p)$, is derived from the cone constraint, equation 4.2 (Also equation 2.2 and figure 2.1.), and additionally uses gradient and boundary information; its construction is detailed in subsection 4.2.1. The compatibility between adjacent surface orientations, $\psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q)$, expresses the smoothness assumption, and is detailed in subsection 4.2.2. Rather than solving the problem directly we use a hierarchical approach, as detailed by Felzenszwalb and Huttenlocher[79], in which we solve the problem at lower resolutions and use the messages of lower resolution levels to initialise higher resolution levels. This reduces the number of iterations required from thousands at full resolution to tens for each level of a hierarchy of halving resolution.

Once belief propagation has converged then equation 4.5 can be used to extract a final FB_8 distribution for each node. For output we require directions rather than distributions, a problem muddied by the ability of the FB_8 distribution to have two maxima with no clear means to choose the correct one due to the concave/convex ambiguity of SfS[2]. A solution to this issue is given in subsection 4.2.3.

4.2.1 Priors on Orientation

For each random variable we have an irradiance value. Using the cone constraint equation, 4.2, and the Bingham-Mardia distribution, given by equation D.15, a distribution can be defined, using the notation of appendix D, as

$$\Omega[2k_i \frac{I}{A}\hat{\mathbf{l}}, -k_i \hat{\mathbf{l}}\hat{\mathbf{l}}^T] \tag{4.6}$$

which captures the cone constraint with a Bingham-Mardia distribution. Its concentration, k_i, is set on the observation that extreme irradiance values, where specularity and non-Lambertian roughness effects can break the Lambertian assumption, are less reliable. Specifically, k_i is linearly interpolated based on its angle with the light source, $\cos^{-1}(I/A)$. These values are set at $0°$, $45°$ and $90°$, with the middle value the highest to reflect its greater certainty. In principle $\psi_p(\hat{\mathbf{x}}_p)$ could be set to equation 4.6 directly, but we multiply this distribution with two others, as detailed in the following.

4.2.1.1 Gradient Information

In the spirit of both Zheng & Chellappa[11] and Worthington & Hancock[21] we use gradient information. Gradient gives an indication of surface orientation, as the surface orientation will usually be on the disc defined to contain both the gradient direction and the direction to the light source. The aforementioned papers assumed a direction rather than a disc, creating a bias to a concave or convex solution. As objects generally have both convex and concave areas this is less than ideal, so we use a disc distribution. Given a normalised gradient direction, $\hat{\mathbf{g}}$, which is in the image plane, $\hat{\mathbf{g}}_z = 0$, we can use the Bingham distribution

$$\Omega[\mathbf{0}, -k_g \hat{\mathbf{d}}\hat{\mathbf{d}}^T], \qquad \hat{\mathbf{d}} = \frac{\hat{\mathbf{g}} \times \hat{\mathbf{l}}}{|\hat{\mathbf{g}} \times \hat{\mathbf{l}}|} \tag{4.7}$$

The concentration parameter, k_g, is set proportional to gradient strength, $|\mathbf{g}|$. Using gradient information is necessary, as without it an arbitrarily rotation of surface normals around the light source direction would have no effect on model probability[3].

4.2.1.2 Gradient Calculation

Rather than a more typical method, such as the Sobel operator, a diffusion method is used to calculate the gradient that is used by equation 4.7. It is robust in the presence of noise and lacks the distortion of methods such as the Sobel operator.

[3] An integration constraint would resolve this, but no method to integrate such information has been found.

Unlike window methods a diffusion approach has the advantage of considering the route to a pixel, and hence giving little weight to data the other side of an edge where data is likely irrelevant. This edge handling is the primary motivation, as in the presence of edges we want the shading gradient of the surface, not the gradient caused by the edge. It is described here first as a random walk, then as it is implemented.

Starting at the pixel in question and being of fixed length each walk contributes a vector going from the walks start to the walks end; the mean of these vectors is the output gradient direction and strength. Every step the walk moves to one of the four adjacent pixels so as to create a walk that tends towards brighter areas.

Given the walk length, w, we have to consider pixels up to w away from the sampled pixel, (x, y), in an iterative approach that takes w steps. Define the probability of a walk being at pixel (u, v) at time step t as $\phi_{t;u,v}$. For the initial time step all walks start at (x, y), $\forall u, v; u = x \wedge v = y \Rightarrow \phi_{0;u,v} = 1, u \neq x \vee v \neq y \Rightarrow \phi_{0;u,v} = 0$. Next define the distribution sum of each pixel as $\xi_{t;u,v} = \phi_{t;u,v}/(4\alpha + \sum_{i,j \in N} I^{\beta}_{i,j})$ where N is the 4-way neighbourhood of the pixel. We can now define $\phi_{t;u,v}$ for non-zero time steps, $\phi_{t;u,v} = (\alpha + I^{\beta}_{u,v}) \sum_{i,j \in N} \xi_{t-1;i,j}$. Once we have got to time step w the gradient direction is found, $\mathbf{g}_{x,y} = \sum_{u,v} \phi_{w;u,v}[u - x, v - y, 0]^T$. Its length is the gradient strength whilst $\hat{\mathbf{g}}$ is the normalised version.

4.2.1.3 Boundary Information

Due to the concave/convex ambiguity[2] we can expect two answers. Bi-modal distributions are output, with the two modes corresponding to the concave and convex interpretations. We use a boundary constraint[17] to bias towards a convex or concave solution. A boundary constraint uses the fact that the edge of an object is tangential to both the viewing direction and the curve of the boundary, which fixes surface orientation. Consideration also has to be made for shadows however, as the edges of shadowed regions are instead tangential to the light source direction and the shadow-edge curve. Boundaries are detected as non-zero pixels adjacent to pixels with a value of zero. We threshold on boundary pixels, a high value implies an object edge, a low value implies a self-shadowing edge. A

Fisher distribution is then multiplied in for such points. We define the indicator variable, b, as 1 on the boundary and 0 elsewhere. The tangential direction is $\hat{\mathbf{t}}$, for object edges it is $-\hat{\mathbf{g}}$, the reversed gradient direction, whilst for shadow edges it is $\hat{\mathbf{l}} \times \hat{\mathbf{d}}/|\hat{\mathbf{l}} \times \hat{\mathbf{d}}|$; hence the distribution is

$$\Omega[k_b b\hat{\mathbf{t}}, \mathbf{0}] \tag{4.8}$$

where k_b is concentration, constant over the image; for a convex bias use a positive value, for concave negative.

Finally, equations 4.6, 4.7 and 4.8 are combined by multiplying them together, with equation D.12, to get the prior for each pixel,

$$\psi_p(\hat{\mathbf{x}}_p) = \Omega[2k_i \frac{I}{A}\hat{\mathbf{l}} + k_b b\hat{\mathbf{t}}, -k_i \hat{\mathbf{l}}\hat{\mathbf{l}}^T - k_g \hat{\mathbf{d}}\hat{\mathbf{d}}^T] \tag{4.9}$$

This is used by equations 4.4 and 4.5 for message passing and belief calculation.

4.2.2 Smoothing

Smoothing assumes that adjacent pixels should have a small angular difference, in preference to a large angular difference. We can express this by setting

$$\psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q) \propto \exp(k_s(\hat{\mathbf{x}}_p^T \hat{\mathbf{x}}_q)) \tag{4.10}$$

where $\psi_{pq}(\hat{\mathbf{x}}_p, \hat{\mathbf{x}}_q)$ is the compatibility between adjacent pixels. This is a Fisher distribution with concentration k_s. Using FB_8 for the messages and dropping equation 4.10 into the message passing equation, equation 4.4, we have

$$m_{p\to q}^t(\hat{\mathbf{x}}_q) = \int_{\mathbb{S}^2} \exp(k_s(\hat{\mathbf{x}}_p^T \hat{\mathbf{x}}_q)) t(\hat{\mathbf{x}}_p) \delta\hat{\mathbf{x}}_p \tag{4.11}$$

$$t(\hat{\mathbf{x}}_p) = \psi_p(\hat{\mathbf{x}}_p) \prod_{u \in (N \setminus q)} m_{u\to p}^{t-1}(\hat{\mathbf{x}}_p) \tag{4.12}$$

Message passing therefore consists of two steps: calculating $t(\hat{\mathbf{x}}_p)$ by multiplying FB_8 distributions together, followed by convolution of the resulting FB_8 distribution by a Fisher distribution, to get $m_{p\to q}^t(\hat{\mathbf{x}}_q)$. The result of this convolution step is

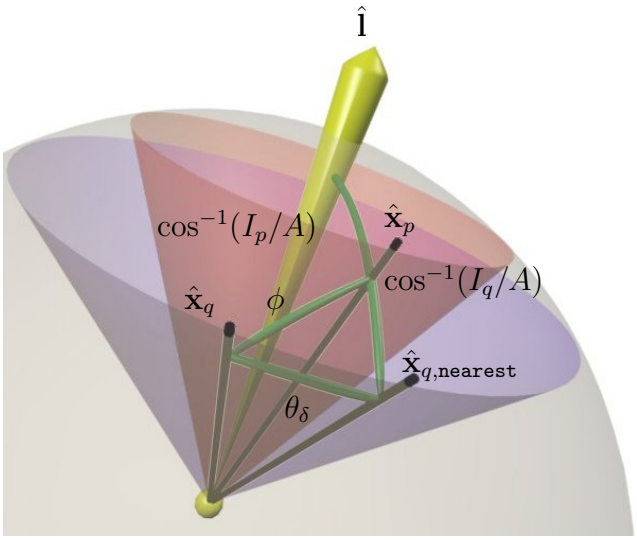


Figure 4.1: Diagram indicating the construction of ϕ. $\hat{\mathbf{l}}$ is the light source direction, whilst the two cones indicate the cone constraints derived for pixel p (red, smaller) and pixel q (blue, larger). Three directions are then shown, $\hat{\mathbf{x}}_p$, $\hat{\mathbf{x}}_{q,\texttt{nearest}}$ and $\hat{\mathbf{x}}_q$. $\hat{\mathbf{x}}_p$ is arbitrary whilst $\hat{\mathbf{x}}_q$ is relative to $\hat{\mathbf{x}}_p$ as required to define ϕ. $\hat{\mathbf{x}}_{q,\texttt{nearest}}$ is the closest $\hat{\mathbf{x}}_q$ can get to $\hat{\mathbf{x}}_p$ whilst fulfilling the cone constraint. These three directions form a triangle on the sphere. The angle $\hat{\mathbf{x}}_p$ to $\hat{\mathbf{x}}_{q,\texttt{nearest}}$ is defined by the cone constraints, $\hat{\mathbf{x}}_{q,\texttt{nearest}}$ to $\hat{\mathbf{x}}_q$ is the parameter θ_δ and the final angle defines ϕ.

not in fact a FB_8 distribution, so approximation as a FB_8 distribution is required. Section 4.3 provides a novel solution to this problem.

Concentration, k_s, still has to be set. Given adjacent surface orientations, $\hat{\mathbf{x}}_p$ and $\hat{\mathbf{x}}_q$, the cone constraint enforces a minimum angle between them, $\cos^{-1}(\hat{\mathbf{x}}_p \cdot \hat{\mathbf{x}}_q) \geq \texttt{abs}(\cos^{-1}(I_p/A) - \cos^{-1}(I_q/A))$. Assuming the surface orientations both comply with the cone constraint then each has a single degree of freedom - their rotation around the light source direction, θ_p and θ_q. The minimum angle is achieved when this rotation is the same for both surface orientations. A difference between these rotation values may be considered, $\theta_p = \theta_q - \theta_\delta$, which results in the angle between the surface orientations being

$$\phi = \cos^{-1}(s_p s_q \cos(\theta_\delta) + c_p c_q) \tag{4.13}$$

where $s_r = \sin(I_r/A)$ and $c_r = \cos(I_r/A)$ for $r \in \{p, q\}$. The offset can be moved to the other cone however, swapping p and q in the above equation; the maximum of these two values is taken. The geometry of calculating ϕ is given in figure

4.1. Given a fixed θ_δ value a user may specify the probability, p, of the surface orientations being within angle ϕ. With θ_δ constant whilst k_s varies the smoothing adapts so the concentration is highest for pixels of similar irradiance but lowest for pixels with greatly differing irradiance.

A method for selecting a Fisher distributions concentration parameter such that it satisfies $P(\cos^{-1}(\hat{\mathbf{x}}_p \cdot \hat{\mathbf{x}}_q) < \phi) = p$ is still needed. ϕ and p are given and the Fisher distribution, equation D.4, is defined by parameter $\mathbf{u}$, where $\mathbf{u} = k_s\hat{\mathbf{x}}_p$, with k_s the concentration parameter that needs estimating. It may be supposed that $\hat{\mathbf{x}}_q$ is measured in spherical coordinates relative to $\hat{\mathbf{x}}_p$, (θ, ψ), where θ is the angle from $\hat{\mathbf{x}}_p$ and ψ is the angle around $\hat{\mathbf{x}}_p$; this makes ψ irrelevant. You may then marginalise equation D.4 away ψ[135, p. 170] (Unlike equation D.4 this is normalised.)

$$P(\theta, k) = \frac{k \exp(k \cos\theta) \sin\theta}{2 \sinh k} \tag{4.14}$$

Finding k_s to fix the probability of being within angle ϕ of the Fisher distributions most probable direction is then a matter of solving

$$\int_0^\phi P(\theta, k_s) d\theta = p \tag{4.15}$$

An analytical solution can not be found, so numerical integration is used. As p is constant a lookup table is constructed from k_s to ϕ, with a binary search and linear interpolation used to calculate the inverse.

4.2.3 Non-probabilistic output

Once run to convergence belief propagation provides per-pixel surface orientation as directional distributions; we need to extract actual surface orientation. The obvious approach of selecting the most likely direction from each distribution independently will not provide a globally consistent solution. Sum-product belief propagation, as used to calculate these distributions, finds marginal distributions which, due to the convex/concave ambiguity[2], have two modes corresponding to these two possibilities. Even with the boundary term bias, equation 4.8, there will still be many nodes where the two modes are equally likely or the wrong

mode is more likely[4]. The solution is to find both maxima for each pixel and then select a consistent set for the entire image. We do this with belief propagation, specifically the min-sum variant which finds the most likely global solution. In this case the solution is a set of choices, between the maxima at each pixel, making for a discrete problem. A method of extracting the two maxima for each pixel is given in section D.4. Given the two maxima, $\hat{\mathbf{x}}_d$, $d \in \{a, b\}$, we have a cost associated with each node, p, for each maxima

$$C_p(d_p) = -\ln(P_{\mathrm{FB}_8}(\hat{\mathbf{x}}_{d_p})) \tag{4.16}$$

where cost is the $-\ln$ of ψ from the belief propagation equations. The cost of adjacent nodes being assigned given directions is

$$C_{pq}(d_p, d_q) = -k_c \hat{\mathbf{x}}_{d_p} \cdot \hat{\mathbf{x}}_{d_q} \tag{4.17}$$

which is a Fisher distribution with concentration k_c. Given equations 4.16 and 4.17 the message passing equation is adapted to become

$$m^t_{p\to q}(d_q) = \xi m^{t-1}_{p\to q}(d_q) + (1-\xi) \min_{m_p} \left(C_{pq}(d_p, d_q) + C_p(d_p) + \sum_{u \in (N \setminus q)} m^{t-1}_{u\to p}(d_p) \right) \tag{4.18}$$

where we have introduced a momentum term, ξ, which is required as otherwise regions can oscillate between a concave and convex interpretation without ever converging. Unlike the hierarchical belief propagation used for determining the distributions, with a fixed number of iterations per level, we run this at full resolution till convergence is detected, defined in terms of the change in messages being sent being below a tolerance[5]. After convergence the orientation assigned to

[4]This occurs due to changes in surface orientation and internal occluding boundaries through which the boundary information does not pass.

[5]Even if optimising this post-processing step were easy it is orders of magnitude faster than the distribution calculation, making such effort hardly worthwhile.

each pixel is the one that minimises

$$C_p(\hat{\mathbf{x}}_{d_p}) + \sum_{u \in N} m_{u \to p}(\hat{\mathbf{x}}_{d_p}) \tag{4.19}$$

4.3 Message Passing

To pass messages using equation 4.11 we have to convolve a FB_8 distribution by a Fisher distribution. After convolution we no longer have a FB_8 distribution, which makes belief propagation intractable. Therefore we approximate the result by a FB_8 distribution. We propose a novel three step procedure to solve this problem[6]:

1. Convert the FB_8 distribution to a sum of Fisher distributions.
2. Convolve each Fisher distribution individually.
3. Refit a FB_8 distribution to the resulting mixture of Fisher distributions.

All three steps involve approximation; in practise this proves to not be a problem, but see subsection 4.3.4 for further consideration of this issue.

4.3.1 Step 1

We approximate the Fisher-Bingham distribution as a sum of un-normalised Fisher distributions. Starting with equation D.14 and separating the right-hand side into two exponential functions you get

$$\exp(\mathbf{v}^T\hat{\mathbf{y}})\exp(\alpha\hat{\mathbf{y}}_x^2 + \beta\hat{\mathbf{y}}_y^2) \tag{4.20}$$

into which we may substitute an approximation of the right-hand multiplier to get

$$\exp(\mathbf{v}^T\hat{\mathbf{y}}) \int_0^{2\pi} \exp(m\hat{\mathbf{y}}_x \cos(\theta) + n\hat{\mathbf{y}}_y \sin(\theta))\delta\theta \tag{4.21}$$

In practise a small number of Fisher distributions will be sampled, rather than the infinite number implied by the integral, to get

$$\exp(\mathbf{v}^T\hat{\mathbf{y}}) \sum_i \exp([m\cos(\theta_i), n\sin(\theta_i), 0]\hat{\mathbf{y}}) \tag{4.22}$$

[6]It is of course trivial, though computationally expensive, to extend this to the FB_8 - FB_8 case.

where the θ_i are equally distributed in $[0, 2\pi)$. This may then be re-written as a sum of Fisher distributions[7]

$$\sum_i \exp((\mathbf{v} + [m\cos(\theta_i), n\sin(\theta_i), 0]^T)^T \hat{\mathbf{y}}) \tag{4.23}$$

m and n need to be determined. To explicitly write the approximation

$$\exp(\alpha\hat{\mathbf{y}}_x^2 + \beta\hat{\mathbf{y}}_y^2) \propto \int_0^{2\pi} \exp(m\hat{\mathbf{y}}_x\cos(\theta) + n\hat{\mathbf{y}}_y\sin(\theta))\delta\theta \tag{4.24}$$

$$\exp(\alpha\hat{\mathbf{y}}_x^2 + \beta\hat{\mathbf{y}}_y^2) \propto 2\pi\mathbf{I}_0(\sqrt{m^2\hat{\mathbf{y}}_x^2 + n^2\hat{\mathbf{y}}_y^2}) \tag{4.25}$$

where $\mathbf{I}_0$ is the modified Bessel function of the first kind, order 0. Whilst similar the two sides of (4.25) are different, and so an exact match is not possible. We may however consider six values of $\hat{\mathbf{y}}$ - $[\pm1, 0, 0]^T$, $[0, \pm1, 0]^T$ and $[0, 0, \pm1]^T$. These vectors are the critical directions of the Bingham distribution (i.e. two minimas, two maximas and two standing points, except in degenerate situations.). Using $[0, 0, \pm1]^T$ we get

$$\exp(0) \propto 2\pi\mathbf{I}_0(0) \equiv 1 \propto 2\pi \tag{4.26}$$

which gives us the constant of proportionality. We can then use $[\pm1, 0, 0]^T$ and $[0, \pm1, 0]^T$ to write

$$\exp(\alpha) = \mathbf{I}_0(\sqrt{m^2}) \qquad \exp(\beta) = \mathbf{I}_0(\sqrt{n^2}) \tag{4.27}$$

which can be rearranged to get values of m and n

$$m = \mathbf{I}_0^{-1}(\exp(\alpha)) \qquad n = \mathbf{I}_0^{-1}(\exp(\beta)) \tag{4.28}$$

This approximation leaves the critical points in the same locations with the same relative values.

[7]Note that they are written here without normalisation terms; to maintain this under a mixture model each Fisher distribution has to be weighted by its inverse normalisation term. This is necessary for the conversion back to a FB_8 to work in step 3.

4.3.2 Step 2

Mardia and Jupp [135, p. 44] give an approximation of the convolution of two von-Mises distributions, i.e. the Fisher distribution but on the circle. The n-dimensional von-Mises-Fisher distribution, of which the Fisher distribution is the $n = 3$ case and the von-Mises distribution is the $n = 2$ case, can be represented as

$$P_{vMF}(\hat{\mathbf{x}}; \hat{\mathbf{w}}, k) \propto \exp(k\hat{\mathbf{w}}^T\hat{\mathbf{x}}) = \psi_n[\hat{\mathbf{w}}, k] \tag{4.29}$$

where $\hat{\mathbf{x}}, \hat{\mathbf{w}} \in \mathbb{R}^n$ and $|\hat{\mathbf{x}}| = |\hat{\mathbf{w}}| = 1$. The approximation given is then

$$\psi_2[\hat{\mathbf{w}}_1, k_1] * \psi_2[\hat{\mathbf{w}}_2, k_2] \approx \psi_2[\hat{\mathbf{w}}_1 + \hat{\mathbf{w}}_2, A_2^{-1}(A_2(k_1)A_2(k_2))] \tag{4.30}$$

where $A_p(k) = \frac{\mathbf{I}_{p/2}(k)}{\mathbf{I}_{p/2-1}(k)}$. This may easily be extended to the Fisher distribution with no angular offset between the distributions

$$\psi_3[\hat{\mathbf{w}}, k_1] * \psi_3[\hat{\mathbf{w}}, k_2] \approx \psi_3[\hat{\mathbf{w}}, A_3^{-1}(A_3(k_1)A_3(k_2))] \tag{4.31}$$

As a computational bonus, $A_3(k)$ is equivalent to the Langevin function

$$A_3(k) = \frac{\mathbf{I}_{1.5}(k)}{\mathbf{I}_{0.5}(k)} = \coth(k) - \frac{1}{k} \tag{4.32}$$

The inverse Langevin function is also needed, its computation can be approximated by its Taylor expansion

$$A_3^{-1}(k) = 3x + \frac{9x^3}{5} + \frac{297x^5}{175} + \frac{1539x^7}{875} + \dots \tag{4.33}$$

4.3.3 Step 3

To derive a Fisher-Bingham distribution from the convolved sum of Fisher distributions we first need the rotational component of the Bingham distribution, which we calculate with principal component analysis [PCA]. The first step is to calculate a weighted mean

$$\bar{\mathbf{m}} = \frac{\sum_i W_i \mathbf{u}_i}{\sum_i W_i} \tag{4.34}$$

$\mathbf{u}_i$ is the indexed Fisher distributions direction vector multiplied by its concentration parameter. W_i is the normalisation term of the indexed Fisher distribution, this will be the normalisation term of the Fisher distribution *after* convolution divided by the normalisation term of the Fisher distribution *before* convolution

$$W_i = \frac{k_{i,\texttt{after}} \sinh(k_{i,\texttt{before}})}{k_{i,\texttt{before}} \sinh(k_{i,\texttt{after}})} \tag{4.35}$$

The second step is to calculate the data matrix, $\mathbf{X}$, followed by extracting the principal components

$$\mathbf{X} = \begin{bmatrix} W_0(\mathbf{u}_0 - \bar{\mathbf{m}}) \\ W_1(\mathbf{u}_1 - \bar{\mathbf{m}}) \\ \vdots \end{bmatrix} \tag{4.36}$$

$$\mathbf{X}^T\mathbf{X} = \mathbf{R}\mathbf{E}\mathbf{R}^T \tag{4.37}$$

$\mathbf{E}$ is the diagonal matrix of eigenvalues; $\mathbf{R}$ is then the rotational component of the Bingham distribution. This works as the FB_8 distribution, when converted into a mixture of Fisher distributions, traces an ellipsoid with the vectors of the Fisher distributions, that is dependent on the Bingham component alone. The directions of greatest variation of this ellipsoid, as found with PCA, then match the axes of the Bingham distribution.

Given six directions and their associated density function values we may fit the remaining parameters to get a distribution with matching probability ratios between the selected directions. Using six instances of[8]

$$\exp(\mathbf{v}^T\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{D}\hat{\mathbf{y}}) = p \tag{4.38}$$

where p and $\hat{\mathbf{y}}$ are known, $\hat{\mathbf{y}} = \mathbf{R}^T\hat{\mathbf{x}}$ and $\mathbf{D}$ is diagonal, we can apply the natural logarithm to both sides to get

$$\mathbf{v}^T\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{D}\hat{\mathbf{y}} = \ln(p) \tag{4.39}$$

[8]It should be noted that equality rather than proportionality is used here. This is irrelevant as multiplicative constants have no effect.

This is a linear set of equations, which can be solved using standard techniques to get $\mathbf{v}$ and $\mathbf{D}$. The final FB_8 distribution is then proportional to

$$\exp((\mathbf{R}\mathbf{v})^T\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\mathbf{R}\mathbf{D}\mathbf{R}^T\hat{\mathbf{x}}) \tag{4.40}$$

These six directions have to be carefully selected to produce a good approximation, as only the sampled directions will be fitted, and the convolved distribution can differ greatly from a FB_8 distribution. The selection strategy used is based on the observation that with no Fisher component the optimal selection of $\hat{\mathbf{y}}$ is $[\pm 1, 0, 0]^T$, $[0, \pm 1, 0]^T$ and $[0, 0, \pm 1]^T$, these being the extremal values of the Bingham distribution[9] (There is also a computational advantage of this selection as they are linearly separable.). Given a Fisher component we may divide through the mixture of Fisher distributions to leave only a (supposed) Bingham component; the estimation procedure will then estimate another Fisher component as well as the Bingham component. This leads to an iterative scheme, where the Fisher component is initialised with the weighted mean of the mixture of Fisher distributions, $\bar{\mathbf{m}}$, and updated after each iteration, by adding the newly estimated Fisher component to the component already subtracted. In practise convergence happens after only two iterations[10]. It should be noted that this approach is the inverse of step 1, the initial conversion to a mixture of Fisher distributions, i.e. if you skip step 2 your output will always match your input, ignoring the error introduced by the use of a finite number of Fisher distributions and numerical approximation.

4.3.4 Analysis of message Passing

A numerical analysis of the convolution approximation gives poor results; just how bad is indicated in figure 4.2. The y-axis is the square root of twice the Kullback-Leibler divergence, as measured in nats. This error measure has an

[9] Optimal is used here in the sense of getting the values right at the extremal points, as ultimately this is about finding the correct modes. If minimising, say, the Kullback-Leibler divergence of the approximation these values would not be optimal.

[10] To accelerate convergence we initialise the Fisher component to the mean of the Fisher vectors, which is already calculated during PCA.

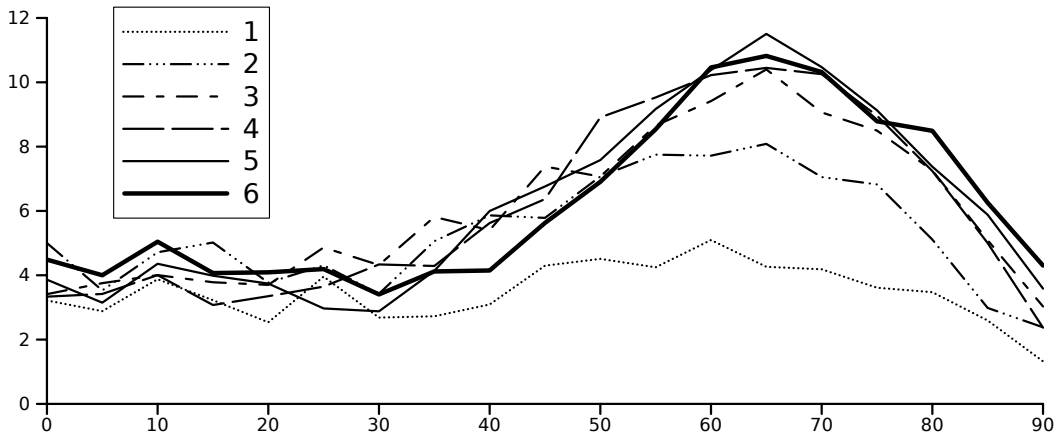


Figure 4.2: This graph shows the error in approximating typical messages of the belief propagation process. The y-axis of this graph shows the square root of twice the Kullback-Leibler divergence as measured in nats, whilst the x-axis gives the angle of a Bingham-Mardia distribution, as set by the irradiance using the cone constraint. Each line represents a different convolution concentration parameter as indicated by the key; a value of 6, as represented by the thick line, is most common in practise. It should be noted that this graph was created stochastically and suffers from heavy noise.

intuitive interpretation, as approximating a Normal distribution with another normal distribution, both with a standard deviation of one, but with an offset between them of these error values gives the same Kullback-Leibler divergence. This leads to the interpretation that approximating a Normal distribution by another normal distribution with 4 standard deviations difference in the mean is comparable to the lower errors shown by the convolution process.

Despite these results the algorithm does work, as the results in section 4.4 show. This can be put down to a key point - the approximation gets the extrema in the right place and with the correct probability ratios, even though it messes up the gaps between them. The approximation used equates $\exp(x)$ and $\mathbf{I}_0(\sqrt{x})$, both of which can be rewritten as power series, such that you are approximating

$$\sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{4.41}$$

with

$$\sum_{n=0}^{\infty} \frac{(0.25x)^n}{(n!)^2} \tag{4.42}$$

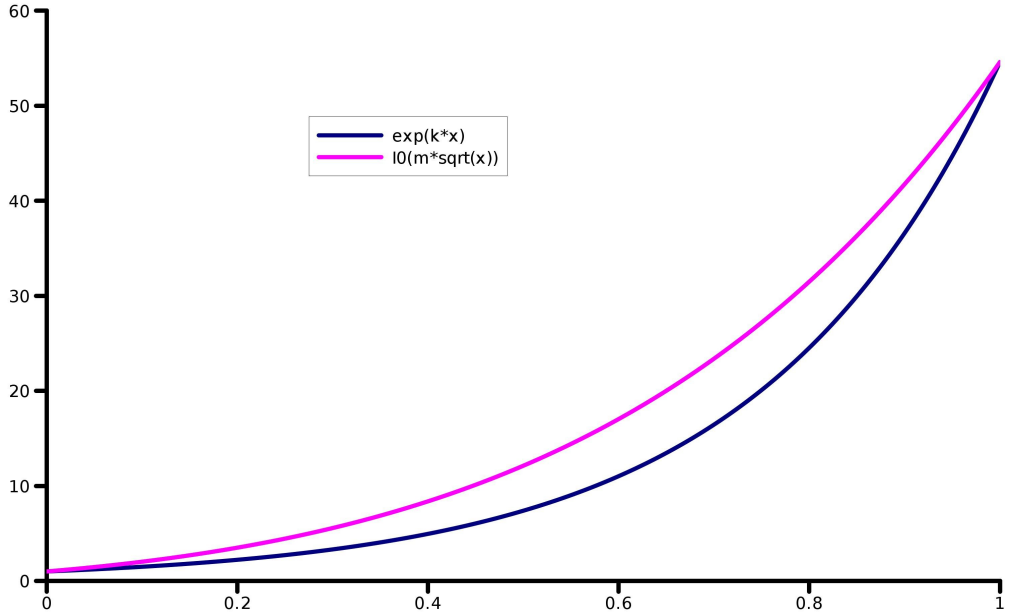
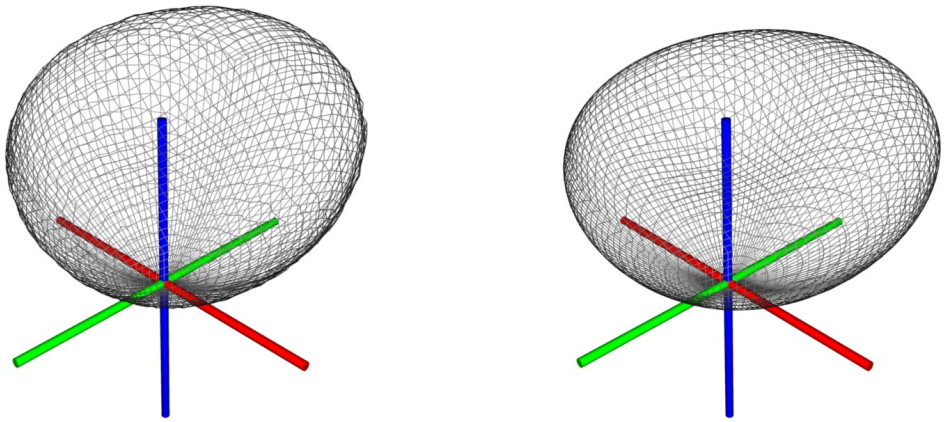


Figure 4.3: Demonstration of the key approximation used by the convolution process. At the left we have a minima, on the right a maxima, that have been matched. The lines then diverge in the centre, indicating the error, which peaks at $x = \cos(45)$.

These are both exponentially shaped functions with parameters set to give equal magnitude at critical points, which is to say the error is zero at the critical points but increases as you move away from them into the space between - this is demonstrated in figure 4.3. As it is the maxima we are interested in the errors are least where the error matters most. Figure 4.4 shows this processes output for an example distribution.



(a) Brute force (b) Approximation

Figure 4.4: This shows convolutions of the distribution represented by figure D.1(d). Brute force shows stochastically calculated ground truth, approximation shows the result of using the approximation of this section.

4.4 Experiments

In this section the presented algorithm is compared to Lee & Kuo[10] and Worthington & Hancock[21], using both synthetic and real data. A more limited comparison is made against Potetz[26]. The first subsection considers the choice of algorithms and testing method. Following this two sub-sections then cover synthetic and real data respectively, past which a section on robustness is given. We conclude on the relative resource usage of the algorithm. To give an executive summary the presented method does badly with oblique lighting and is relatively weak with synthetic data, but for real world input it is consistently ahead. It is also the fastest algorithm, though not the most memory efficient.

4.4.1 Choices

There are three choices to be made when comparing algorithms - which data sets to use, which algorithms to compare against, and the methods of comparison. The existence of a good review within a field will provide these answers, but as covered in subsection 2.1.1 neither of the existing reviews is sufficient for this.

Zhang et al.[9] in their '99 review provide a synthetic data set, which is now standard. Whilst the synthetic data set has been used their real world test set has no ground truth, and has images with complex lighting and materials, sufficiently far from the assumptions made by SfS algorithms to make a qualitative analysis dubious at best. We have therefore provided our own real world test set with ground truth; the hope is that it will also be widely used.

Competitor selection is unfortunately a matter of pragmatism - even a review paper can only cover a limited set. We select Lee & Kuo as, whilst we disagree with the testing approach, the '99 review paper concluded it to be the best, and so we consider it to be representative of the best of earlier works. Of the more recent algorithms we have Worthington & Hancock and its variants, and include the original in the testing. The various improvements are relatively minor and so it can be reasonably treated as indicative of future work. This leaves two notable recent algorithms - the viscosity based approach of Prados et al[41] and the belief propagation approach of Potetz[26]. Prados et al. are excluded as they require lighting falloff to be visible in the image, and hence are solving a different problem. We have no working implementation of Potetz[26] and are hence limited to comparing the single test case given in his paper; in figure 4.17 we allow this by using the same data.

Finally we consider the method of quantitative comparison. Subsection 2.1.1 discuses the flaws inherent with a depth based error metric - we therefore use surface orientation error[11]. This choice avoids penalising large scale errors - this makes sense as it is the small scale detail that SfS is expected to extract[5]. When integration is a separate step this also avoids including it in the error measurement. Whilst taking average angular error is obvious averages have problems - a few

[11] There is also differential of depth ($\delta z/\delta x$ and $\delta z/\delta y$) error, but this unjustifiably weights errors for viewer-oblique orientations higher than for viewer-facing orientations.

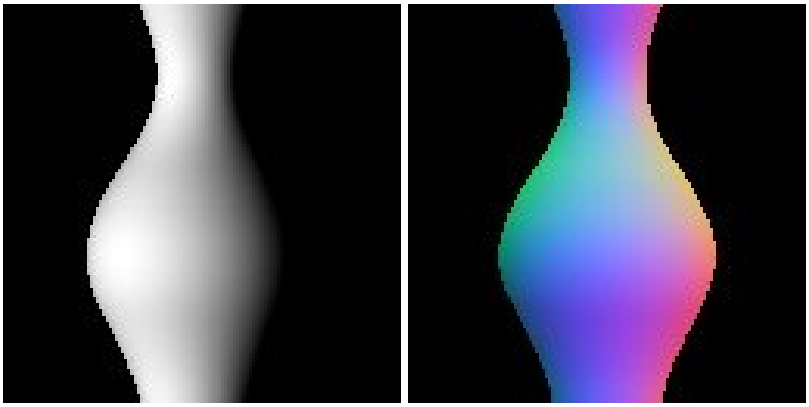
bad results can greatly raise the error of an otherwise good algorithm. Robust statistics can resolve this, but introduces parameters with no obvious choice of value. Instead we use *inlier percentage errors*, giving the percentage of inliers for various outlier thresholds. The key advantage is in distinguishing between an algorithm that gets a lot of good answers, but also some bad answers, and an algorithm that is mediocre, but avoids bad answers.

4.4.2 Synthetic

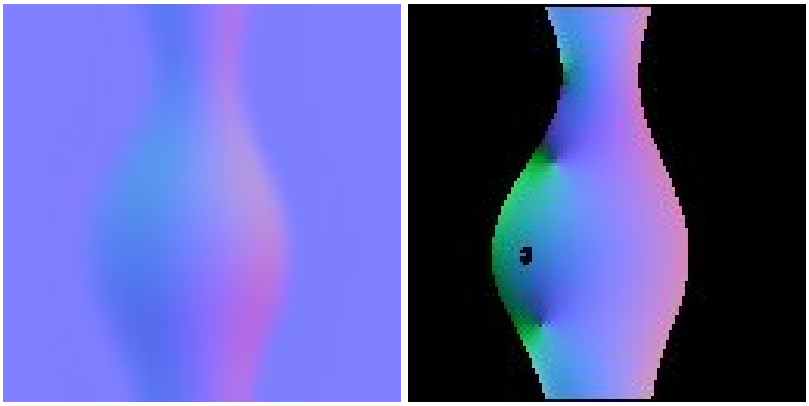
Figures 4.5,4.6,4.7 and 4.8 give the four synthetic inputs used. The two meshes, obtained from the paper by Zhang et al.[9], each have two renderings, the 45° version with the light source at $[-\sqrt{(}2), 0, \sqrt{(}2)]^T$ and the 90° version with the light source at $[0, 0, 1]$; using a right handed coordinate system. None of the algorithms do particularly well, despite this task being relatively easy due to the inputs satisfying all the assumptions made. Talking qualitatively Lee & Kuo suffers from blurring and a lack of detail, but does not make any large mistakes, whilst Worthington & Hancock has areas where it does very well but also a number of artefacts and regions where it has failed. This large number of errors makes integrating the Worthington & Hancock output to generate a $3D$ model unreliable. All approaches have issues with the 45° images, but our proposed method is especially bad, and fails almost entirely on the Mozart 45° input. For this reason we also present a run with alternative parameters, that better handle the oblique lighting, at the expense of real world performance. This alternate version handles the Mozart images much better, and shows an ability to extract features sharply, without the artefacts present in Worthington & Hancock. The Vase outputs are relatively reasonable, but the algorithms suffer from various flaws, such as the singularities in both Worthington & Hancock and the presented algorithm[12].

Figure 4.9 gives the quantitative results for the synthetic inputs. Sticking to the 90° images, where the light is at $[0, 0, 1]^T$, Lee & Kuo is left behind, though

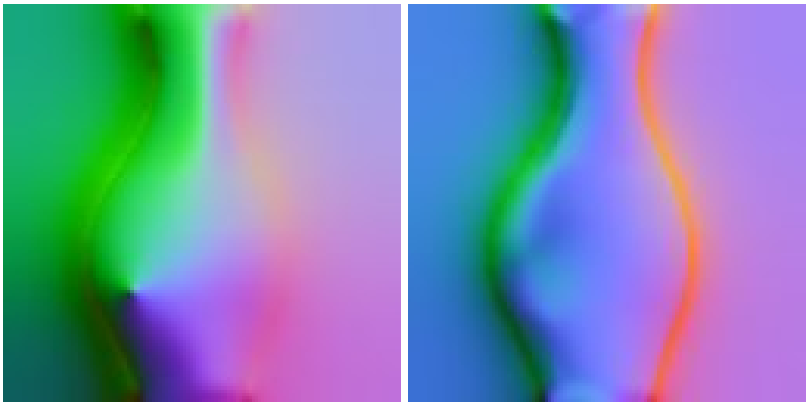
[12]The cause of these singularities is unknown. To speculate it seems that the algorithms prefer a large error for a single pixel, rather than error distributed over the local region, in certain cases. For the presented the use of the Bingham-Mardia distribution supports this theory as it generates an error equivalent to the difference of the cosine of the angles, which for certain angles means the error from summing the angles is less than the sum of the individual errors.



(a) Vase $45°$ (b) Ground truth

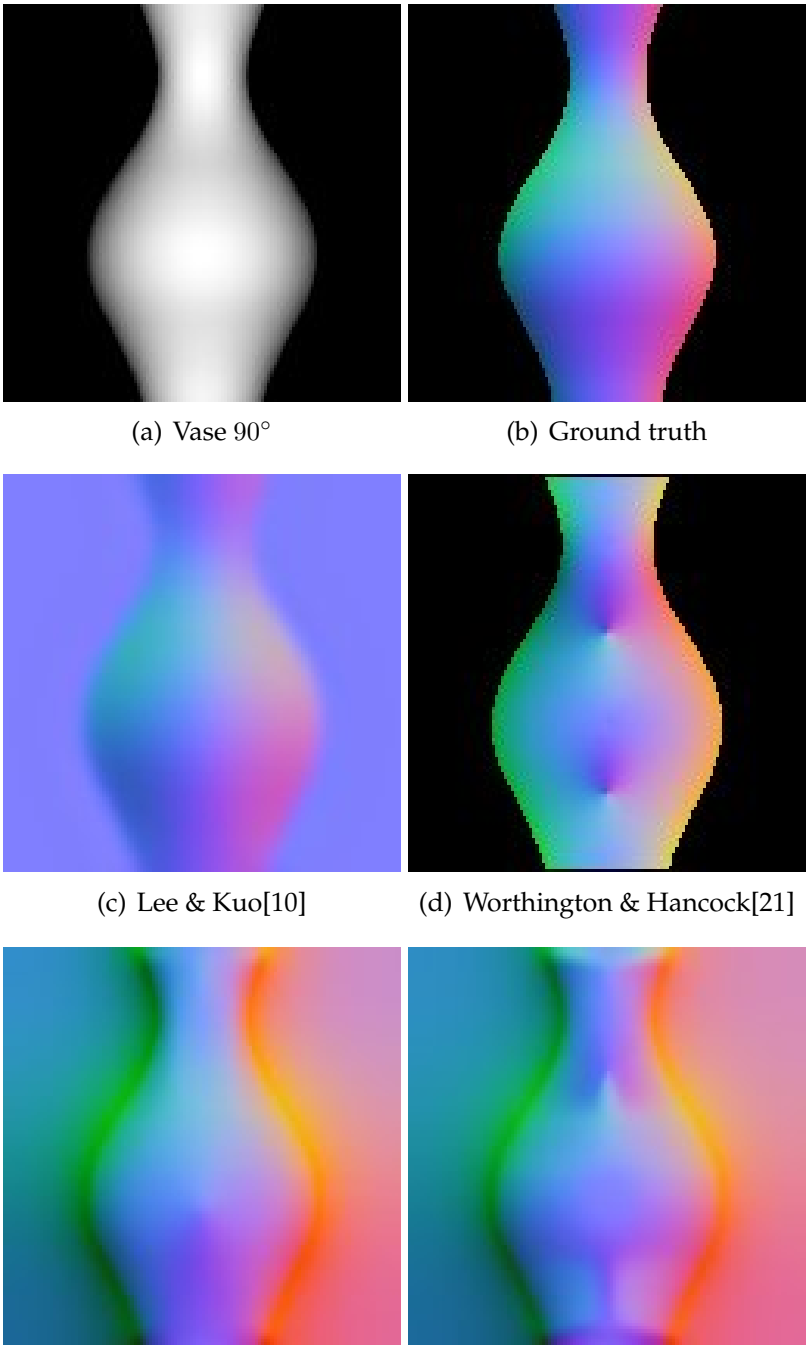


(c) Lee & Kuo[10] (d) Worthington & Hancock[21]



(e) Presented algorithm (f) Presented algorithm, alternate parameters

Figure 4.5: Synthetic input vase $45°$ with ground truth and outputs as labelled. Red is x, green y and z blue, as taken from the surface orientation normals. Red and green are mapped to $[-1, 1]$, blue to $[0, 1]$.

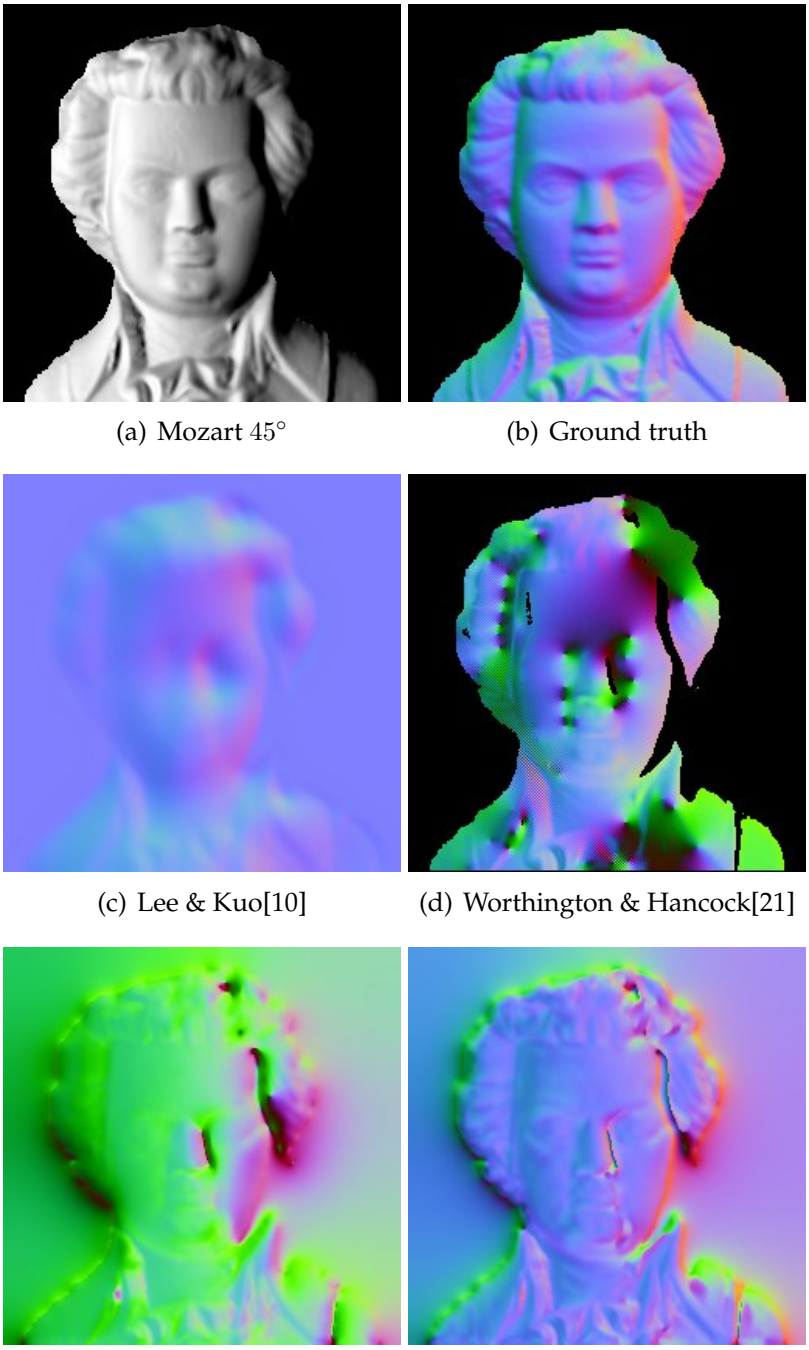


(a) Vase 90° (b) Ground truth

(c) Lee & Kuo[10] (d) Worthington & Hancock[21]

(e) Presented algorithm (f) Presented algorithm, alternate parameters

Figure 4.6: Synthetic input vase 90°; see figure 4.5 and text for details.

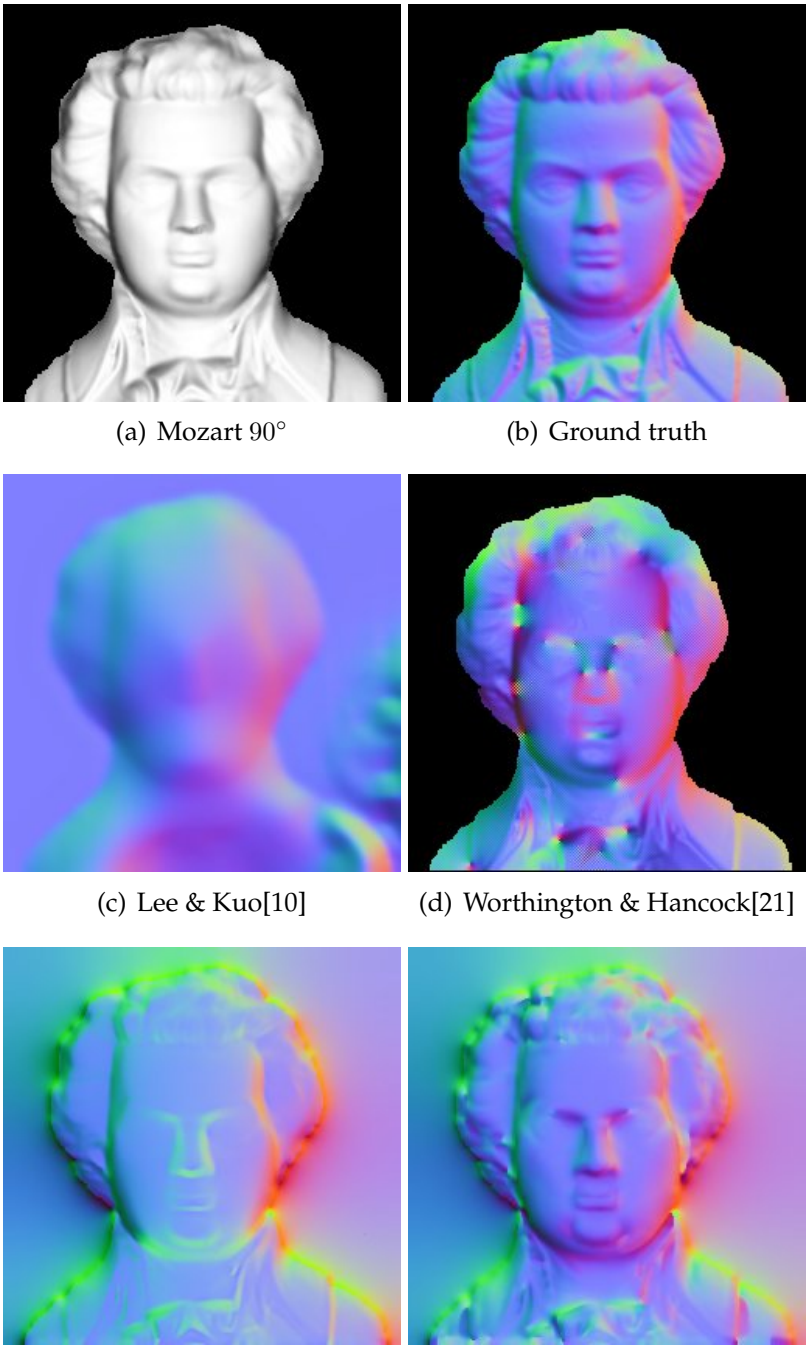


(a) Mozart $45°$ (b) Ground truth

(c) Lee & Kuo[10] (d) Worthington & Hancock[21]

(e) Presented algorithm (f) Presented algorithm, alternate parameters

Figure 4.7: Synthetic input Mozart $45°$; see figure 4.5 and text for details.



(a) Mozart 90°

(b) Ground truth

(c) Lee & Kuo[10]

(d) Worthington & Hancock[21]

(e) Presented algorithm

(f) Presented algorithm, alternate parameters

Figure 4.8: Synthetic input Mozart 90°; see figure 4.5 and text for details.

the numbers do show its ability to avoid large errors as it consistently overtakes Worthington & Hancock for the larger error thresholds. Worthington & Hancock appears to have an advantage at the very lower ends of the scale, this is presumably because it perfectly matches the irradiance information, unlike the others. Passed a certain error threshold however both parameter sets for the presented algorithm take the lead. Moving to the 45° inputs, where the light source direction vector is $[-\sqrt{2}, 0, \sqrt{2}]^T$, things do not go well, and the presented algorithm using the original parameters is trounced by the competitors. The alternative parameters fix this, but they do so by increasing certainty in the irradiance information and using stronger smoothing terms and boundary information, which make these alternate parameters poor for real world input.

4.4.3 Real

Figure 4.10 gives real world inputs for testing against; 3D renders of the results from the three algorithms, alongside ground truth, are then given in figures 4.12,4.13,4.14 and 4.15. Quantitative analysis is given in figure 4.16, identically to the synthetic results. The input images were captured in a dark room using a camera with a calibrated response curve, at the same time 3D models were recovered with a Cyberware 3030 head scanner; ground truth normal maps were then produced, see figure 4.11. Albedo values were calculated using the ground truth data - an albedo estimate was generated for every pixel and the median taken for robustness. Unlike the synthetic inputs these objects only approximately obey the assumptions, making them a harder problem to solve. This is especially true for the Bard and Head images, as they have dirty surfaces, whilst Venus and Sunev[13] are derived from a freshly spray painted, and therefore clean, bust. However, the Sunev image is noticeably harder than the rest due to its fine detail and bumpy shape, which limits the usefulness of strong smoothing and boundary constraints.

Quantitatively the presented algorithm is consistently ahead, except for small error thresholds where it is occasionally matched by Worthington & Hancock.

[13]Venus, backwards, on the grounds its the back of the Venus bust.

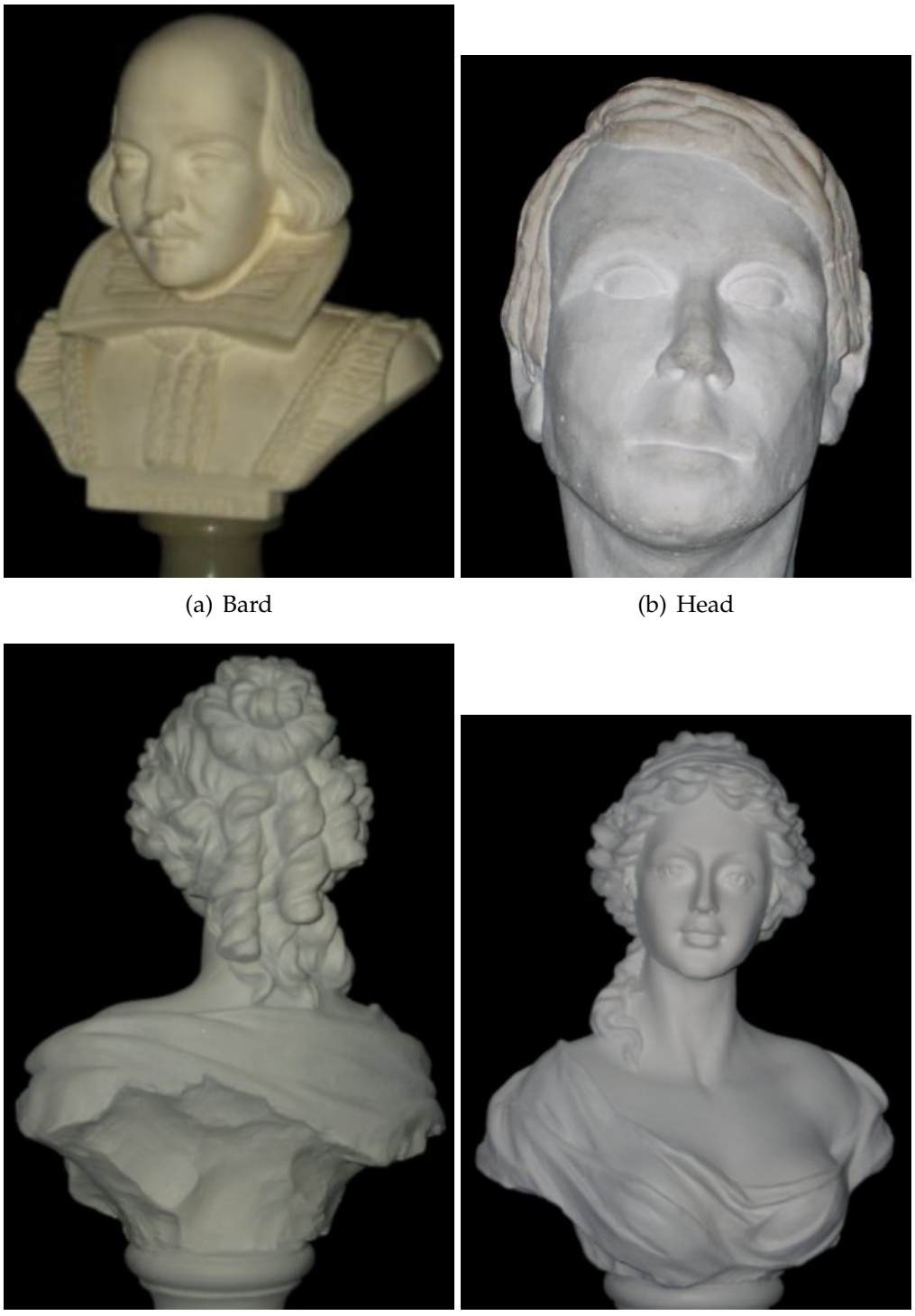
Vase 45°	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.5	1.6	2.8	3.8	5.5	14.2	28.6	55.8	71.4	77.8
Worthington & Hancock[21]	2.6	5.7	8.6	11.3	14.6	24.9	32.6	39.9	49.2	61.8
Presented algorithm	0.2	0.7	1.8	3.4	5.4	22.0	38.1	54.6	69.9	78.7
Presented algorithm, alt.	0.3	1.1	2.8	5.7	9.2	24.6	42.3	60.8	77.3	90.2

Vase 90°	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.7	3.0	7.2	13.8	20.2	66.7	79.3	87.1	92.8	97.2
Worthington & Hancock[21]	2.9	6.3	9.4	11.9	14.4	25.0	34.2	41.7	48.4	54.7
Presented algorithm	1.3	5.1	13.3	22.1	35.0	80.7	89.8	92.4	94.1	95.9
Presented algorithm, alt.	0.3	0.8	3.4	20.5	28.7	55.8	70.2	79.8	87.9	93.5

Mozart 45°	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.2	0.7	1.5	2.5	3.8	16.1	35.0	54.7	67.2	76.1
Worthington & Hancock[21]	1.3	3.2	4.8	6.3	8.0	15.3	23.0	30.7	37.7	44.4
Presented algorithm	0.2	0.5	1.0	1.5	2.0	5.7	10.0	14.7	19.9	25.5
Presented algorithm, alt.	0.1	0.9	2.7	6.7	15.8	42.6	59.5	70.1	77.7	83.8

Mozart 90°	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.1	0.5	1.3	2.3	3.8	17.4	36.2	52.3	66.4	76.0
Worthington & Hancock[21]	3.3	7.3	11.3	15.3	19.6	34.9	47.5	57.0	64.3	70.7
Presented algorithm	0.4	1.4	3.3	6.2	10.1	30.7	48.9	65.5	75.5	82.7
Presented algorithm, alt.	0.5	1.8	4.4	11.3	20.0	52.7	70.2	80.4	87.8	91.8

Figure 4.9: Synthetic results. Each grid gives results for an input image, as from figures 4.5, 4.6, 4.7 and 4.8. Each row gives results for a specific algorithm. Each column gives the percentage of pixels within a given error bound, i.e. the $< 1°$ row gives the percentage of pixels where the estimated surface orientation is within 1 degree of ground truth. The percentage is only for pixels where ground truth is available.

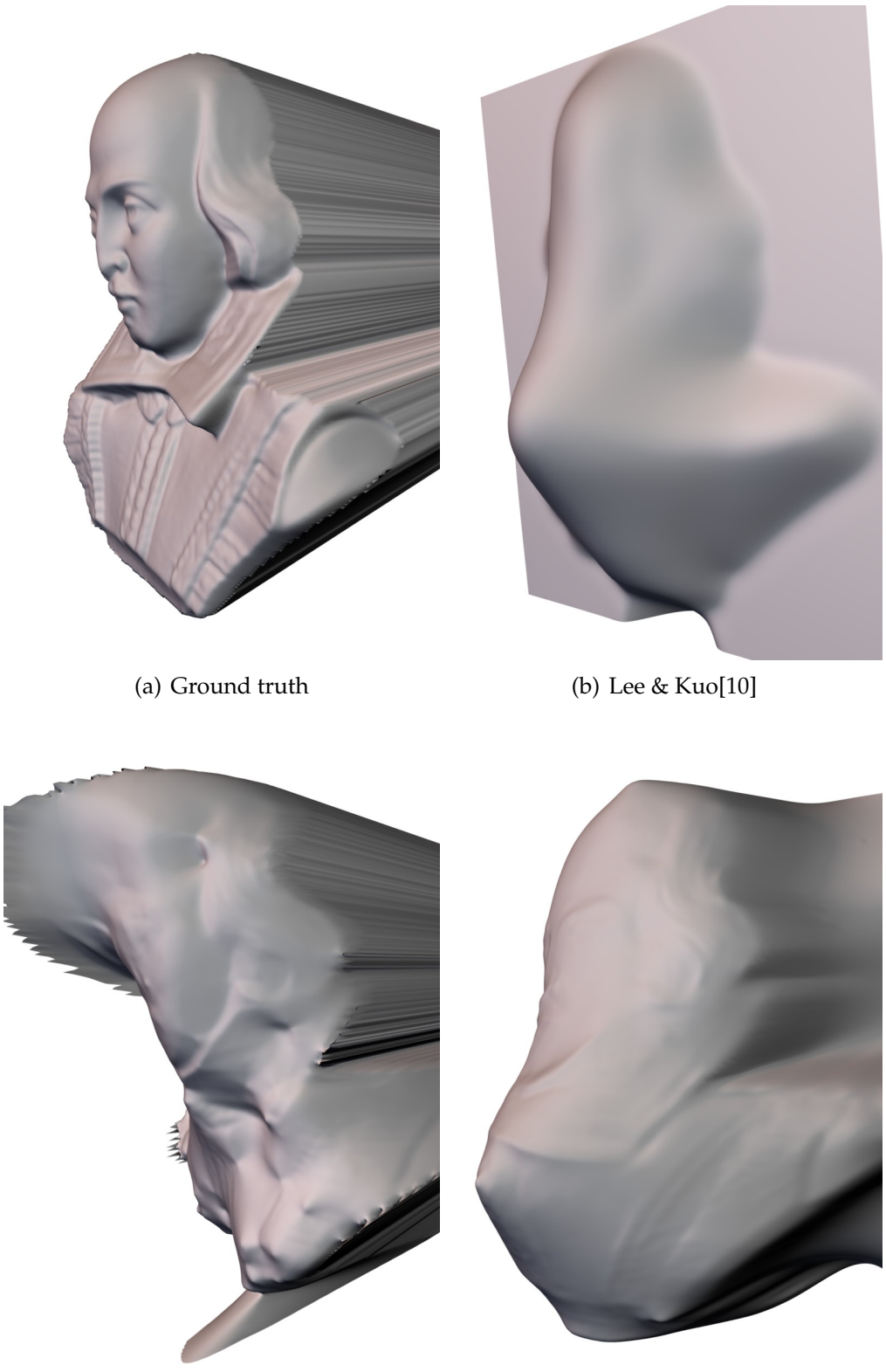


(a) Bard (b) Head

(c) Sunev (d) Venus

Figure 4.10: Real inputs. The one light source is the cameras flash, giving a light source direction vector of $[0, 0, 1]^T$. For reference, the albedos are 0.619 for bard, 0.743 for head, 0.587 for Sunev and 0.645 for Venus.

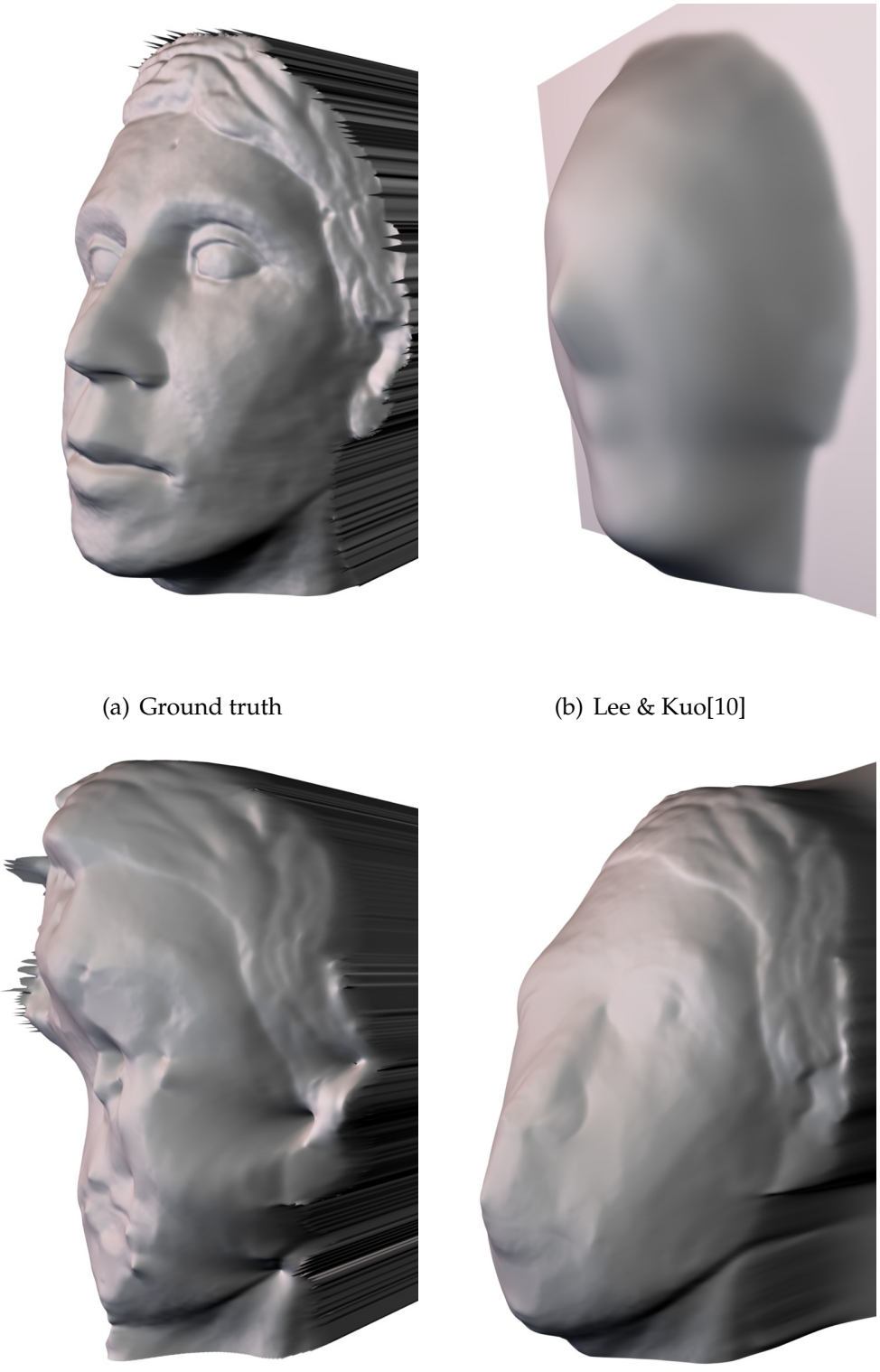
Figure 4.11: The ground truth normal maps for the four input images, as calculated using a head scanner. For calculating the error only areas where data is present are used, the black masked out regions are either background or where the scanner failed due to occlusion.



(a) Ground truth (b) Lee & Kuo[10]

(c) Worthington & Hancock[21] (d) Presented algorithm

Figure 4.12: Results for the bard input with ground truth. Note that the ground truth render was generated by integrating the ground truth normal map, rather then using the mesh directly, for fairness.



(a) Ground truth

(b) Lee & Kuo[10]

(c) Worthington & Hancock[21]

(d) Presented algorithm

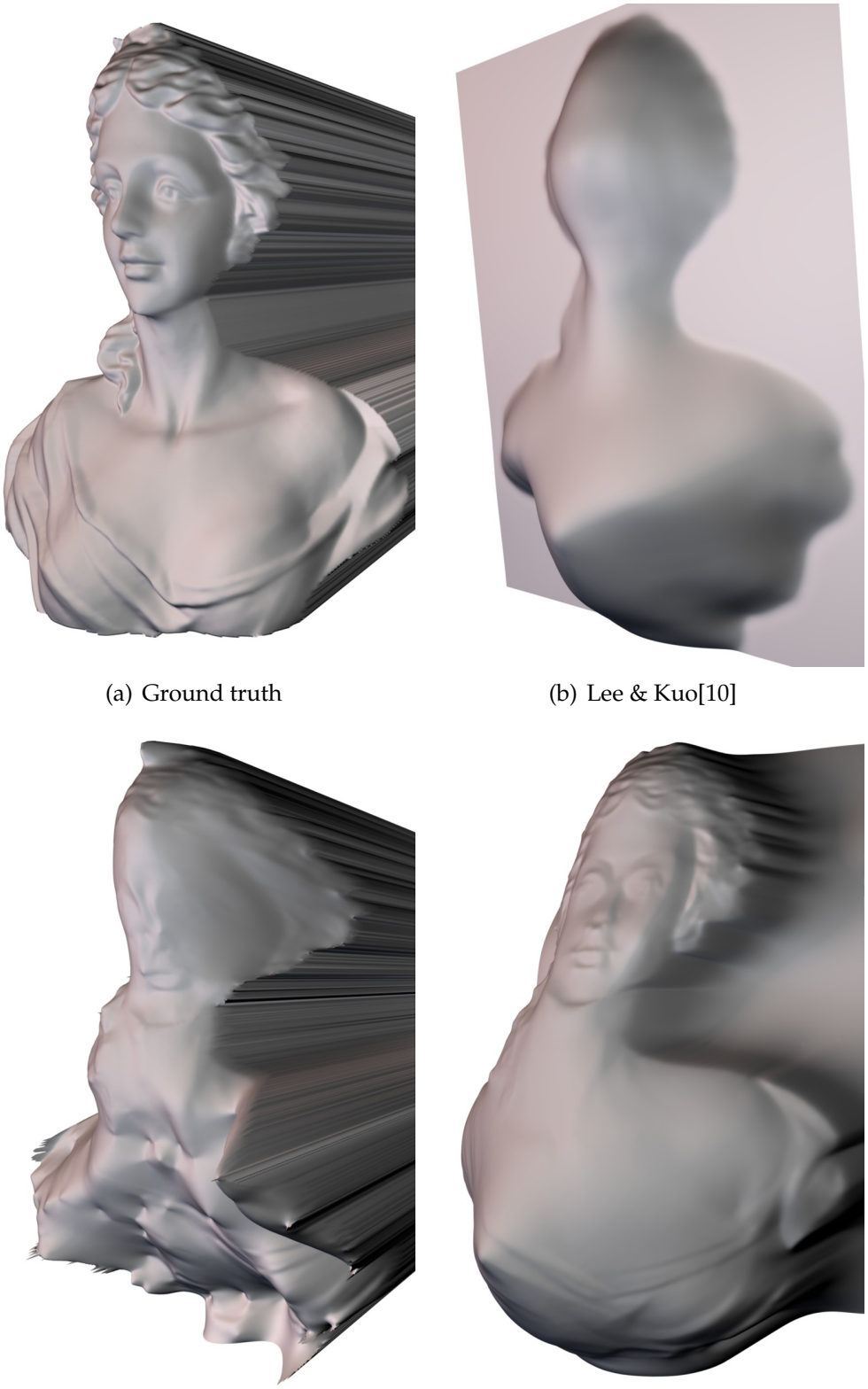
Figure 4.13: Results for the head input with ground truth.

(a) Ground truth

(c) Worthington & Hancock[21]

(d) Presented algorithm

Figure 4.14: Results for the sunev input with ground truth. For this input Lee & Kuo fails.



(a) Ground truth

(b) Lee & Kuo[10]

(c) Worthington & Hancock[21]

(d) Presented algorithm

Figure 4.15: Results for the Venus input with ground truth.

Bard	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.0	0.3	0.6	1.2	1.9	8.1	19.4	27.5	33.9	41.4
Worthington & Hancock[21]	0.1	0.5	1.1	1.9	2.9	9.0	15.2	22.3	30.1	37.5
Presented algorithm	0.0	0.5	1.1	2.0	2.9	10.7	18.2	25.3	32.2	39.4

Head	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.3	1.1	2.3	3.8	5.7	18.8	34.1	47.1	58.8	68.7
Worthington & Hancock[21]	0.1	0.7	1.4	2.6	4.0	13.6	25.3	38.6	51.7	61.5
Presented algorithm	0.5	1.9	4.2	7.3	11.1	33.8	49.8	62.2	72.2	79.0

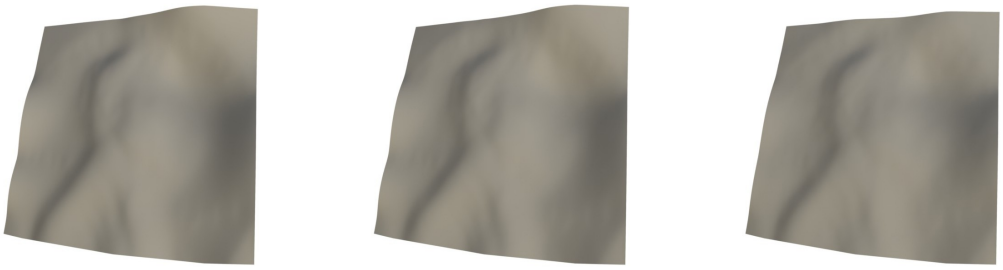
Sunev	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.2	0.7	1.5
Worthington & Hancock[21]	0.1	0.4	0.8	1.5	2.2	7.7	15.2	23.7	32.0	39.3
Presented algorithm	0.2	0.6	1.3	2.2	3.4	13.5	27.7	41.8	54.2	62.3

Venus	$< 1°$	$< 2°$	$< 3°$	$< 4°$	$< 5°$	$< 10°$	$< 15°$	$< 20°$	$< 25°$	$< 30°$
Lee & Kuo[10]	0.0	0.4	0.8	1.5	2.4	10.6	23.6	36.1	48.4	59.8
Worthington & Hancock[21]	0.1	0.5	1.1	1.8	2.7	9.2	16.8	24.4	32.9	40.9
Presented algorithm	0.1	0.5	1.1	2.1	3.4	14.5	28.9	42.1	53.1	62.3

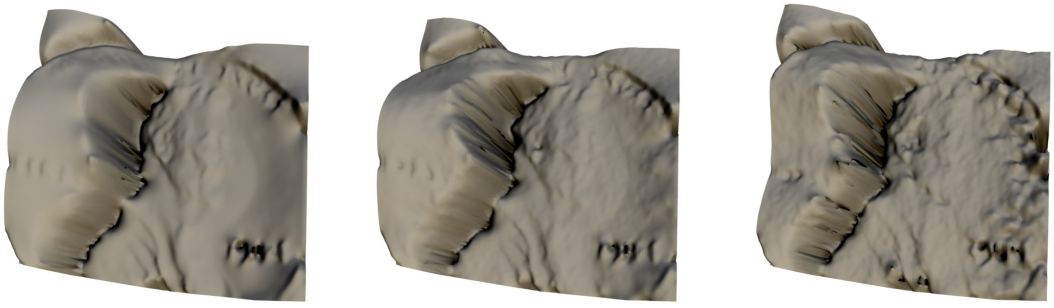
Figure 4.16: Real results. Each grid gives results for input images from figure 4.10. See figure 4.9 for explanation of grids.

This is in contrast to the synthetic results, and shows how effective a probabilistic approach is when dealing with real world data, i.e. a probabilistic approach will generally handle poor assumptions better; it also highlights the risk of testing synthetic data alone. Looking at the output renders qualitatively the quantitative results are not surprising. Qualitatively, Lee & Kuo is, as for the synthetic data, overly smooth and lacking detail, whilst Worthington & Hancock has little semblance of the input image detectable. The presented algorithm is clearly not perfect, but it gets the broad shape right and at the same time preserves fine details. Venus is a good example of this, as whilst the face lacks depth it is certainly there, with eyes, nose and hair all sharply visible.

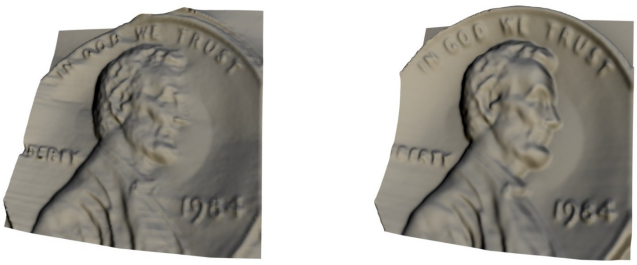
(a) Input



(b) Lee & Kuo[10]



(c) Presented algorithm



(d) Potetz (left) & ground truth (center)

Figure 4.17: The first three rows give an indication of the noise handling of the presented algorithm as well as Lee & Kuo's algorithm. Each column is for a different noise level, the first being none, the second with the standard deviation set to 32, and the third with it set to 64. The final row shows the Potetz result followed by ground truth. The Potetz render is comparable to the renders directly above it.

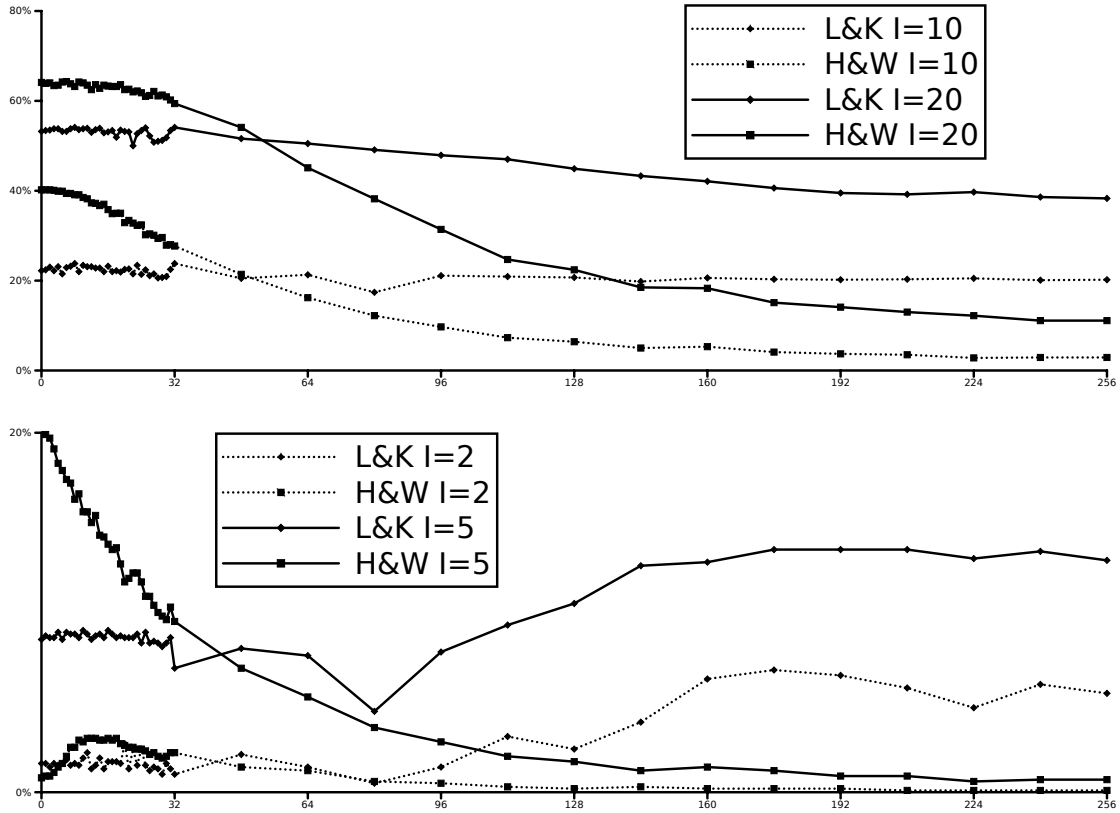


Figure 4.18: Graphs indicating the error rate as noise is increased for the presented algorithm (H & W) and Lee & Kuo (L & K). The y-axis indicates the percentage of inliers where an inlier is within 20° or within 10° for the top graph and within 5° or 2° for the bottom graph. The x-axis is the standard deviation of additive iid Gaussian noise.

4.4.4 Robustness

We have tested Lee & Kuo and the presented method on increasing independent and identically distributed Gaussian noise to compare how they fail. For the test data we used the penny image[9] lit with a light source direction of $[\sqrt{2}, 0, \sqrt{2}]$ - we are using the presented methods alternate parameters. Worthington & Hancock has been excluded as the lack of boundary information causes it to fail. This is the same image used by Potetz[26], and so a direct comparison is possible[14]. Figure 4.17 shows the input and outputs for no noise and two levels of noise. Lee & Kuo is typically blurred, which gets more blurred as the noise increases. The presented method suffers from a bad case of curl - the shadow on the back of Lincoln's head blocks information transfer and forces the relative depths of the two regions to

[14]Thanks go to Potetz for providing his data, as seen in figure 4.17.

	lk	wh	hw
Vase 45° (128x128)	14m,32s	2m,36s	28s
Vase 90° (128x128)	15m,9s	2m,47s	26s
Mozart 45° (256x256)	1h,7m	15m,45s	1m,42s
Mozart 90° (256x256)	1h,9m	15m,51s	1m,43s
Bard (212x270)	8h,54m	44m,18s	1m,33s
Head (328x381)	20h,4m	1h,50m	3m,14s
Sunev (319x472)	22h,42m	1h,53m	4m,17s
Venus (346x457)	21h,22m	1h,46m	4m,31s

Figure 4.19: Run times for all the outputs presented in figures 4.9 and 4.16. The algorithm abbreviations are *lk* for Lee & Kuo, *wh* for Worthington & Hancock and *hw* for the presented algorithm. Input resolution is provided alongside the names, see text for further details.

be calculated using the thin strip of pixels above his head, where the word 'we' indicates a large change of depth, leading to the error seen. Lee & Kuo and Potetz both have an integration constraint, which stops this happening. Ignoring this the presented algorithm has far more detail than Lee & Kuo. Comparing with Potetz is a lost cause however - the Potetz algorithm has even managed to extract the text well enough to be read, though the presented is on the edge of readability for the last word.

Considering the robustness test in figure 4.18 the inlier percentage is graphed as noise increases, for various inlier thresholds. The presented is better than Lee & Kuo for low noise, with an asymptotic falloff as noise increases. It crosses Lee & Kuo roughly at a standard deviation of 48. Lee & Kuo is unusual in converging to a flat line, which for low inlier percentages is better than the low noise level. This is because as noise increases instead of trying to fit the noise, as the presented algorithm does, Lee & Kuo converges to a flat plane, which is a reasonable coin representation. This is certainly preferable behaviour. Finally, it should be noted that this test is limited - iid Gaussian noise is not indicative of the systemic error of approximating reality by constant-albedo Lambertian-shaded surfaces.

4.4.5 Time & Memory

Figure 4.19 gives runtime information for the algorithms[15]. It is potentially misleading however as both competitor algorithms require a fixed set of iterations be set, as detecting convergence for both is tricky. These were set to 20000 iterations for synthetic input and 150000 iterations for real input for Lee & Kuo; and 10000 iterations for synthetic and 25000 for real for Worthington & Hancock. Whilst these numbers are higher than need be they are not more than five times that required to get a reasonable answer, and two times that required to get answers of the sort given in the quantitative analyses. Lee & Kuo would probably get marginally better results if given more iterations still. The point is still made however that the presented algorithm is five times faster in the worst case, and 745 times in the best case than these competitor algorithms, with the gap increasing as resolution is increased.

In regards to memory consumption the roles are reversed however. Lee & Kuo stores only depth for each pixel during runtime, it also has to keep its hierarchy in memory simultaneously, but this still means it consumes less than $4/3$ floats per pixel. Worthington & Hancock on the other hand stores a single surface orientation per pixel, making it 3 floats per pixel (This could easily be reduced to 2 floats as the vectors are normalised.). The presented algorithm uses a checker-board update pattern to half memory consumption, but still needs to store all four incoming messages for each pixel, with each message being the 12 floats of a FB_8 distribution; this makes for 48 floats per pixel altogether. Whilst it has a hierarchy it only needs to store the level it is currently working on.

A comparison with Potetz lacks precision, due to not having precise numbers or knowledge of the hardware used, but referring back to subsection 2.1.2.2 Potetz quotes a runtime of 'several hours' and memory consumption of 6-10KB per pixel. Whilst Potetz clearly gets better results the presented algorithm does very well in comparison, considering the comparatively minuscule runtime and memory consumption used - the penny image takes just under 30 seconds with the presented algorithm.

[15]Run on a Core 2 Duo 2Ghz processor with 1GB of RAM, all implementations single threaded.

4.5 Conclusions

Previous sections have presented a new algorithm for solving the classical shape-from-shading problem. It is a probabilistic approach that makes use of belief propagation with the eight parameter Fisher-Bingham distribution - a key contribution is the procedure for smoothing a field of FB_8 distributions within a BP framework, as this can be applied to other problems. A post-processing step, also using BP, resolves ambiguity in the probabilistic output when choosing a point estimate. Results shows better handling of real input than has been previously presented[16]. It also runs to completion extremely quickly. Additional consideration should be made of the ability to provide this algorithm with arbitrary prior distributions - this allows tight integration with algorithms that provide orientation information. For instance, a stereopsis algorithm could provide a prior on surface orientation.

There are a number of issues that we will now discuss however:

- Not having an integration constraint is almost certainly the greatest weakness. Providing one is no simple task however, and is likely to change the nature of the algorithm entirely. It would make the use of gradient information redundant however. Possible approaches include:
  - A simple method is to run the algorithm and then use an integration algorithm that can handle the curl. An option would exist to then feedback the post-integration directions into the algorithm and run again, with a bias towards such directions in the hope it would be biased towards a better solution. Such an approach would be slow and lack justification however.
  - In principle a probability distribution could be found that includes the integration constraint - this would be a distribution over at least three surface orientations simultaneously with a mathematically inconvenient constraint between them. If such a distribution could be constructed then BP could be run with many such distributions overlapping, with

[16] Potetz[26] would probably do better, but published results only show it with synthetic input.

equality terms between shared surface orientations. Its questionable if this could be done practically, but if it could then the potential would be great.

  - Probability distributions over depth could be introduced, and linked up. This would make this algorithm somewhat similar to Potetz, and make things much more complicated.

  - The post processor could select the highest probability integrable surface, rather than the highest probability surface with curl, which is then separately integrated to remove the curl. In practise this approach is probably as hard as the initial SfS problem however.

  It is the use of an integration constraint that allows Potetz[26] to do better, but, equally, the integration constraint will typically be computationally intensive as its probability distribution is a Dirac function - this approaches speed can in no small part be attributed to its lack of an integration constraint.

- Sum-product belief propagation is currently used - in principal using min-sum BP would get an answer directly, skipping the post processing step. The FB_8 distribution does not lend itself to such an approach however, and another distribution, probably nonparametric, would be needed.

- Oblique lighting is a definite problem. The source of this issue has to be the bias terms introduced via the prior using gradient and boundary information - it is the only term that considers the direction to the viewer; precisely how this causes the observed problems is unclear however. Considering the ultimate aim is to combine this SfS approach with stereopsis, where such information is not needed, this is actually a low priority problem within the scope of the complete work.

- Integration with stereopsis is the ultimate goal. The algorithm already lends itself to this scenario, but an omission exists as the approach of the previous chapter requires Gaussian distributions on change in disparity, whilst this method provides surface orientation in the form of FB_8 distributions - a

conversion would need to be made. Such a conversion will have to be necessarily approximate, as these two representations are entirely different.

Chapter 5
Light Source Estimation

THIS chapter considers further improvement to the base SfS & stereopsis algorithm of chapter 3; specifically a light source estimation module. As presented in chapter 3 the approach requires the light source direction be provided by the user. This is tedious information to obtain, and excludes the approach from wholly automated use, unless there are extra sensors or suitable calibration objects in the scene. The previous work drives the current scenario - we want to estimate an infinitely distant point light source as this is the required input to the SfS & stereopsis algorithm. Lambertian reflectance is assumed. Similarly, the input is a calibrated stereo pair, to which stereopsis can be applied to obtain depth. As covered in section 2.4 a lot of light source estimation algorithms assume known shape, which stereopsis in principle provides. Practically however stereopsis makes mistakes that are often large and systemic, especially in regions of constant albedo. Regions of constant albedo are required for an algorithm to work (justified below), as albedo has to be assumed unknown. Consequentially, stereopsis is most likely to give a bad answer in the areas that matter most to estimating light source direction. This makes this a particularly hard problem, and one that has not previously been tackled[1].

A probabilistic approach is taken, designed to be robust to the many outliers that do not conform to the Lambertian reflectance and piecewise constant albedo assumptions. The formulation is given next, followed by a discussion of how to get distributions on surface orientation from a stereo pair; following that section 5.3 manipulates the equations into something that can be sensibly optimised.

[1]Evidently a 3D mesh can be created using stereopsis and fed into many of the algorithms in subsection 2.4; the errors are so great that this will not work however.

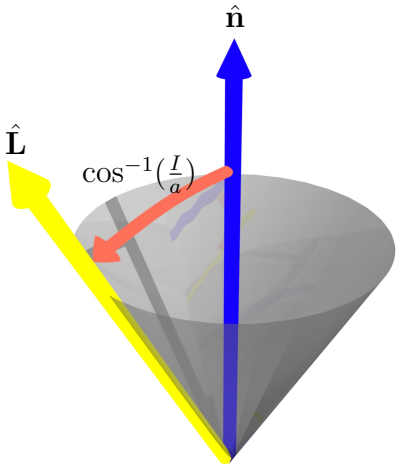


Figure 5.1: The geometric interpretation of the Lambertian reflectance information, with the light source direction as the variable rather than the more typical surface orientation.

Implementation details are given in section 5.4; section 5.5 gives results and analysis, and, finally, section 5.6 gives conclusions.

5.1 Formulation

As for the previous chapters Lambertian reflectance with a single light source is assumed; it is again restated, in a form suitable for this problem[2]:

$$I_{x,y} = a_{x,y}(\hat{\mathbf{n}}_{x,y} \cdot \hat{\mathbf{l}}) \tag{5.1}$$

where $I_{x,y}$ is irradiance, $a_{x,y}$ is albedo, which can vary between pixels, $\hat{\mathbf{n}}_{x,y}$ surface orientation and $\hat{\mathbf{l}}$ the direction to the light source. In a reversal of the previous chapter irradiance and surface orientation are considered known whilst the albedo and light source direction are unknown. The geometric observation of Worthington & Hancock[21], the cone constraint, may be reversed, making light source direction the variable, as shown in figure 5.1. With this interpretation if we know the albedo of a pixel and its surface orientation then the light source direction is constrained to a cone; it then only takes two or three pixels to constrain the direction to the light source. Unfortunately albedo is unknown and surface orientation known unreliably. Assuming constant albedo for the entire image is not an option, as stereopsis needs variable albedo for correspondence to work.

Disparity from a calibrated stereo pair provides depth which can then be differentiated to obtain surface orientation. As indicated this is problematic - firstly, differentiation of a noisy input amplifies the noise; secondly, the disparity map as estimated by stereopsis is not only noisy but will make systemic errors due to bad correspondence and the use of smoothing terms. Quantisation affects are common. As the estimated surface orientation is unreliable we consider a directional probability distribution over surface orientation, $P_{x,y}(\hat{\mathbf{n}}_{x,y})$, rather than a point estimate. Switching to negative log-likelihood we may now define a cost function over an image, M, in terms of surface orientation

$$C_M = \sum_{x,y \in M} -\ln(P_{x,y}(\hat{\mathbf{n}}_{x,y})) \tag{5.2}$$

The problem is therefore to select a light source direction to minimise this cost;

[2]See equation B.6 for more detail.

this of course requires a relationship between surface orientation and the light source direction, which is provided by the Lambertian reflectance equation / cone constraint. A hard constraint is therefore introduced

$$\forall x, y \in M; \frac{I_{x,y}}{a_{x,y}} = \hat{\mathbf{n}}_{x,y} \cdot \hat{\mathbf{l}} \tag{5.3}$$

which brings the light source direction into the problem by rewriting equation 5.1; it also brings the unknown albedo.

Albedo being unknown is the key problem - by varying the albedo of each pixel any light source direction may satisfy the constraint, at least for surface orientations not facing away from the light source. This leaves no choice but to consider the relationship between the unknown albedo values of multiple pixels. Specifically, we choose to consider regions of constant albedo. The following is adaptable to scenarios where ratios are known, however constant albedo is far more common in real world scenes, and finding constant albedo areas via segmentation is reasonable. By assuming that a group of pixels with varying irradiance and surface orientation share a common albedo value some information in regards to the light source direction is provided. Typically this is a very flat distribution, only indicating a slightly reduced probability for a small subset of directions; however, by combining the distributions from many constant albedo regions in the image an estimate may be made. Given that every pixel belongs to a segment, $S(x, y)$, and that each segment has an albedo value, $a_{S(x,y)}$, the constraint may be rewritten

$$\forall x, y \in M; \frac{I_{x,y}}{a_{S(x,y)}} = \hat{\mathbf{n}}_{x,y} \cdot \hat{\mathbf{l}} \tag{5.4}$$

which is applied when minimising equation 5.2.

5.2 Steropsis to Orientation

Probabilistic representation of surface orientation requires a directional distribution, such distributions being covered in appendix D. Specifically, a Fisher distribution is used, see section D.2,

$$P(\hat{\mathbf{n}}) \propto \exp(k\hat{\mathbf{u}} \cdot \hat{\mathbf{n}}) \tag{5.5}$$

where the x, y subscript has been dropped for neatness. Selection of the Fisher distribution is made due to it being symmetric and decreasing as you move away from the direction of highest probability, $\hat{\mathbf{u}}$. This makes the treatment of the next section both tractable and computationally reasonable.

Using the Fisher distribution requires estimating a concentration parameter, k, and direction, $\hat{\mathbf{u}}$, for each pixel. Stereopsis does not estimate surface orientation in such a way that a Fisher distribution is even close to correct, and most algorithms only provide a point estimate anyway. Specifically, a variant of Felzenszwalb & Huttenlocher[79] is used, as documented in subsection 3.3.1. Being a discrete algorithm, and quantisation being highly problematic, the result is smoothed before use. This is done by fitting a Gaussian to each entry - the mean is taken to be the given disparity value, and the standard deviation is calculated using a bi-square m-estimate of scale[136, p. 34] applied to the DSI, the DSI being defined per-pixel using Luv difference[3]. Gaussian belief propagation as discussed in section 3.2 is then used to smooth the result. The inter-pixel smoothing variances are modulated on the dissimilarity between pixels; specifically using a sigmoid applied to Euclidean difference in Luv colour space.

This stereopsis approach provides a point estimate of disparity, so we assume that it is either approximately right or entirely wrong. The approximately right scenario can be handled with the Fisher distribution, by setting $\hat{\mathbf{u}}$ from the disparity map. This calculation involves using two view geometry from appendix A to get

[3]It should be noted that an alternate method, using the Laplace approximation, is used in subsection 3.3.1. The presentation order is in fact reversed from the chronological order - this m-estimate came first. These two methods give broadly similar results - the change is primarily motivated by the Laplace approximation being much faster and having less parameters to tune. It is also simpler to implement.

points in $3D$ space for each pixel, followed by forward differencing, which is done by constructing triangles with neighbouring points and using the triangles normal as $\hat{\mathbf{u}}$. The wrong scenario is handled later in the formulation using cost caps, to limit the influence of (the many) bad pixels.

Concentration, k, also needs to be set. An obvious approach would be to directly use the stereopsis matching cost to provide costs for specific directions, i.e. turning triples of costs for the three disparities of forward differencing into directions, with the associated costs summed to assign a cost to the direction. The problem here is the number of samples quickly becomes too great - considering ± 4 around the stereopsis selected value provides for $(2 \times 4 + 1)^3 = 729$ samples, and ± 4 is not really large enough. Any such procedure applied to every pixel in an image will take too long[4]. Instead a two step procedure is taken - Gaussian distributions on disparity are assigned to all pixels and a concentration parameter derived from them. The Gaussian fitting is achieved exactly as for the smoothing step. A bivariate Gaussian can then be constructed representing the change in disparity in x and y. Given that we represent a univariate Gaussian by

$$\mathcal{N}[\mu, \sigma^2](x) \propto \exp\left(-\frac{1}{2}\frac{(x-\mu)^2}{\sigma^2}\right) \tag{5.6}$$

we have three variables involved in forward differencing - $\mathcal{N}_{x,y}[\mu_a, \sigma_a^2]$, $\mathcal{N}_{x+1,y}[\mu_b, \sigma_b^2]$ and $\mathcal{N}_{x,y+1}[\mu_c, \sigma_c^2]$. Forward differencing therefore requires calculating the bivariate Gaussian distribution represented by

$$\mathcal{N}[\mu, \Sigma](\Delta_x, \Delta_y) \propto \int_{-\infty}^{\infty} \mathcal{N}[\mu_a, \sigma_a^2](d)\mathcal{N}[\mu_b, \sigma_b^2](d + \Delta_x)\mathcal{N}[\mu_c, \sigma_c^2](d + \Delta_y)\delta d \tag{5.7}$$

where the Δ_n terms are the differentials of disparity in the respective directions. The input distributions have been parametrised in terms of differentials and a disparity value, d, which is then marginalised away; simplified it is a bivariate

[4]This was actually tried - it gave run times over an order of magnitude greater than running every other step in the entire algorithm. Time was primarily split between the fitting procedure (A version of Fisher[137] adjusted with bias terms for the scenario at hand.), and the SVD operations required to triangulate the disparities into 3D space.

Gaussian

$$\mathcal{N}[\mu, \Sigma](\Delta_x, \Delta_y) \propto \exp\left(-\frac{1}{2}(\mathbf{z}-\mu)^T\Sigma^{-1}(\mathbf{z}-\mu)\right) \tag{5.8}$$

which works out to have the parameters

$$\mathbf{z} = \begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} \tag{5.9}$$

$$\mathbf{u} = \begin{bmatrix} \mu_b - \mu_a \\ \mu_c - \mu_a \end{bmatrix} \tag{5.10}$$

$$\Sigma^{-1} = \begin{bmatrix} \frac{\rho_b^2}{\rho_a+\rho_b+\rho_c} + \rho_b & \frac{\rho_b\rho_c}{\rho_a+\rho_b+\rho_c} \\ \frac{\rho_b\rho_c}{\rho_a+\rho_b+\rho_c} & \frac{\rho_c^2}{\rho_a+\rho_b+\rho_c} + \rho_c \end{bmatrix} \tag{5.11}$$

where ρ_n is the inverse variance, also known as the precision,

$$\rho_n = \frac{1}{\sigma_n^2} \tag{5.12}$$

Converting a Gaussian on disparity change into a Fisher distribution is not possible, instead we proceed to calculate a concentration parameter by matching regions of equal probability around the modes of the Gaussian and Fisher distributions, noting that both are uni-modal. Whilst not ideal such a procedure is tractable and results in concentration increasing sensibly as disparity estimates become more certain, which is as desired. Given the bivariate Gaussian an ellipse can be calculated such that a sample from the distribution has a specific probability of being inside. This ellipse is equivalent to a multiplier of the Mahalanobis distance defined by the covariance matrix, which as the specific probability is a fixed parameter of the algorithm can be calculated once. The calculation may be done using the inverse of the cumulative chi square distribution function, calculated by numeric integration. Given the ellipse we have five points - its centre and the points on the edge that intercept its major and minor axes. All five points can be converted into orientations. An issue exists as points imply differences in disparity without providing an actual base disparity to be a difference from - in an orthographic projection this would not matter, but with perspective it does. This is resolved

by using the disparity output by stereopsis as the base; this is then equivalent to a local orthographic assumption around each pixels depth estimate. With this assumption the centre point is the maximal direction of the Fisher distribution, $\hat{\mathbf{u}}$. The angles between this direction and the other directions can then be taken. Each given angle indicates a concentration parameter as it indicates how much probability should be found within that angle of $\hat{\mathbf{u}}$ - a conversion from angle to concentration has already been defined by solving equation 4.15 in subsection 4.2.2 for the SfS smoothing. Given the four angles provided by the ellipse a single angle has to be selected - the maximum is taken. Whilst a mean etc. could also be used the maximum tends to under estimate the concentration, which is preferred to avoid bad data being too influential.

5.3 Simplification

The cost equation, 5.2, with the cone constraint, equation 5.4 is not suitable for direct optimisation. A sequence of modifications are now applied to produce a form which can be conveniently optimised, the actual optimisation then being the subject of the next section. These manipulations are primarily a consequence of specifically using the Fisher distribution, and of some of the variables being irrelevant to determining light source direction only.

5.3.1 Merging the cone constraint

Surface orientation is not relevant to the answer so it is removed from the optimisation using the cone constraint. This is advantageous as the resulting cost no longer requires a separate cone constraint, as it is merged in; an unconstrained minimisation is an easier problem to solve. Using negative log-likelihood the per-pixel cost function, from equation 5.5, is

$$C_p(\hat{\mathbf{l}}, a_S, \hat{\mathbf{n}}) = -k\hat{\mathbf{u}} \cdot \hat{\mathbf{n}} + c \tag{5.13}$$

where c is the negative log of the distributions normalisation term; it is now ignored as costs are only compared against each other, making it irrelevant. Subscripting of variables by x, y has been pruned. Light source direction, $\hat{\mathbf{l}}$, and segment albedo, a_S, are currently unused and have been included in anticipation of the the next step, where $\hat{\mathbf{n}}$ is minimised over whilst complying with the cone constraint, which is dependent on $\hat{\mathbf{l}}$ and a_S.

The Fisher distribution only depends on the angle between $\hat{\mathbf{n}}$ and $\hat{\mathbf{u}}$, with the lowest cost found when this angle is minimised. Equally, the cone constraint requires that the angle between $\hat{\mathbf{n}}$ and $\hat{\mathbf{l}}$ be fixed, to restate

$$\frac{I}{a_S} = \hat{\mathbf{n}} \cdot \hat{\mathbf{l}} \tag{5.14}$$

This occurs at the direction on the cone derived from the Lambertian constraint that is closest to $\hat{\mathbf{u}}$, which means that $\hat{\mathbf{n}}$ must be in the plane shared by $\hat{\mathbf{l}}$ and $\hat{\mathbf{u}}$; $\hat{\mathbf{n}}$

may therefore be expressed as

$$\hat{\mathbf{n}} = \alpha\hat{\mathbf{l}} + \beta\hat{\mathbf{u}} \tag{5.15}$$

where α and β choose which unit vector in the plane is used. Inserting this into the cone constraint, equation 5.14, gets

$$\frac{I}{a_S} = (\alpha\hat{\mathbf{l}} + \beta\hat{\mathbf{u}}) \cdot \hat{\mathbf{l}} \tag{5.16}$$

$$\frac{I}{a_S} = \alpha + \beta\hat{\mathbf{u}} \cdot \hat{\mathbf{l}} \tag{5.17}$$

As $\hat{\mathbf{n}}$ is of unit length its dot product with itself should equal 1

$$1 = (\alpha\hat{\mathbf{l}} + \beta\hat{\mathbf{u}}) \cdot (\alpha\hat{\mathbf{l}} + \beta\hat{\mathbf{u}}) \tag{5.18}$$

$$1 = \alpha^2 + \beta^2 + 2\alpha\beta\hat{\mathbf{u}} \cdot \hat{\mathbf{l}} \tag{5.19}$$

the definition of surface orientation, equation 5.15, can now be inserted into the cost equation, 5.13, to get

$$C_p(\hat{\mathbf{l}}, a_S) = -k\hat{\mathbf{u}} \cdot (\alpha\hat{\mathbf{l}} + \beta\hat{\mathbf{u}}) \tag{5.20}$$

$$C_p(\hat{\mathbf{l}}, a_S) = -k\alpha\hat{\mathbf{u}} \cdot \hat{\mathbf{l}} - k\beta \tag{5.21}$$

Given the two constraints on α and β above, equations 5.17 and 5.19, simultaneous equations can be used to complete this equation. The first constraint, equation 5.17, can be rearranged to express α in terms of β,

$$\alpha = \frac{I}{a_S} - \beta\hat{\mathbf{u}} \cdot \hat{\mathbf{l}} \tag{5.22}$$

Inserting equation 5.22 into equation 5.19, and rearranging removes α and hence provides an expression for β

$$\beta^2 = \frac{1 - \frac{I^2}{a_S^2}}{1 - (\hat{\mathbf{u}} \cdot \hat{\mathbf{l}})^2} \tag{5.23}$$

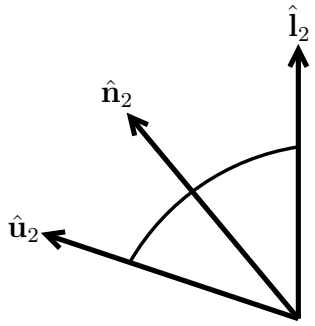


Figure 5.2: The problem of finding the minimum cost whilst satisfying the Lambertian constraint can be solved in 2D, due to the use of a symmetric distribution on surface orientation. See text for details.

Doing the same again and inserting equation 5.22 into the cost equation, 5.21, to remove α, gets

$$C_p(\hat{\mathbf{l}}, a_S) = -k\frac{I}{a_S}\hat{\mathbf{u}}\cdot\hat{\mathbf{l}} + k\beta(\hat{\mathbf{u}}\cdot\hat{\mathbf{l}})^2 - k\beta \tag{5.24}$$

$$C_p(\hat{\mathbf{l}}, a_S) = -k\frac{I}{a_S}\hat{\mathbf{u}}\cdot\hat{\mathbf{l}} - k\beta(1 - (\hat{\mathbf{u}}\cdot\hat{\mathbf{l}})^2) \tag{5.25}$$

into which we may insert equation 5.23 to remove β

$$C_p(\hat{\mathbf{l}}, a_S) = -k\frac{I}{a_S}\hat{\mathbf{u}}\cdot\hat{\mathbf{l}} - k\frac{\sqrt{1-\frac{I^2}{a_S^2}}}{\sqrt{1-(\hat{\mathbf{u}}\cdot\hat{\mathbf{l}})^2}}(1 - (\hat{\mathbf{u}}\cdot\hat{\mathbf{l}})^2) \tag{5.26}$$

$$C_p(\hat{\mathbf{l}}, a_S) = -k\frac{I}{a_S}\hat{\mathbf{u}}\cdot\hat{\mathbf{l}} - k\sqrt{1-\frac{I^2}{a_S^2}}\sqrt{1-(\hat{\mathbf{u}}\cdot\hat{\mathbf{l}})^2} \tag{5.27}$$

which is the per-pixel cost function with the cone constraint integrated and $\hat{\mathbf{n}}$ removed.

Formulating the previous using $3D$ vectors is quite complicated. An arguably simpler formulation is to use the observation that the problem is really $2D$, as indicated by equation 5.15, and project the entire problem onto a plane and use $2D$ vectors. Figure 5.2 shows this, where the subscript 2 has been used to indicate the $2D$ equivalents. Light source direction is arbitrarily set to be

$$\hat{\mathbf{l}}_2 = [1, 0]^T \tag{5.28}$$

Using this and on account of the cone constraint $\hat{\mathbf{n}}_2$ can only be in two positions on a $2D$ plane such that it makes the angle $\cos^{-1}(I/a_S)$ with $\hat{\mathbf{l}}_2$. This fixes it to

$$\hat{\mathbf{n}}_2 = \left[\frac{I}{a_S}, \pm\sqrt{1 - \frac{I^2}{a_S^2}} \right]^T \tag{5.29}$$

which makes use of the Pythagorean identity, $sin^2\theta + \cos^2\theta = 1$. Using the same principle, that the angle between $\hat{\mathbf{l}}_2$ and $\hat{\mathbf{u}}_2$ is $\cos^{-1}(\hat{\mathbf{u}} \cdot \hat{\mathbf{l}})$, obtains

$$\hat{\mathbf{u}}_2 = \left[\hat{\mathbf{u}} \cdot \hat{\mathbf{l}}, \pm\sqrt{1 - (\hat{\mathbf{u}} \cdot \hat{\mathbf{l}})^2} \right]^T \tag{5.30}$$

From these definitions the per-pixel cost equation may be rewritten by inserting the above into equation 5.13[5]

$$C_p(\hat{\mathbf{l}}, a_S) = \frac{-kI\hat{\mathbf{u}} \cdot \hat{\mathbf{l}}}{a_S} - k\sqrt{1 - \frac{I^2}{a_S^2}}\sqrt{1 - (\hat{\mathbf{u}} \cdot \hat{\mathbf{l}})^2} \tag{5.31}$$

which is identical to equation 5.27, calculated with the $3D$ vector method. This is equation 5.13 minimised with respect to $\hat{\mathbf{n}}$, and now a function of $\hat{\mathbf{l}}$ and a_S due to the integration of the cone constraint, equation 5.14,

$$\underset{\hat{\mathbf{n}}}{\operatorname{argmin}}(C_p(\hat{\mathbf{l}}, a_S, \hat{\mathbf{n}})) = C_p(\hat{\mathbf{l}}, a_S) \tag{5.32}$$

5.3.2 Cost Refinement

The per-pixel cost function, equation 5.31, can be re-written as

$$C_p(\hat{\mathbf{l}}, a_S) = \varpi r + \varphi\sqrt{1 - r^2} \tag{5.33}$$

where

$$\varpi = -k\hat{\mathbf{u}} \cdot \hat{\mathbf{l}} \qquad \varphi = -k\sqrt{1 - (\hat{\mathbf{u}} \cdot \hat{\mathbf{l}})^2} \tag{5.34}$$

[5]The signs of the square roots have been set identical; this guarantees that the minimum cost is found.

and $r = I/a_S$. If we presume that $\hat{\mathbf{l}}$ is constant whilst a_S is varying such that $r \in [0, 1]$ then we have a convex function with a single minima to be found when

$$r = \frac{|\varpi|}{\sqrt{\varpi^2 + \varphi^2}} \tag{5.35}$$

However, when r is larger than 1, or $I > a_S$, the function fails, as it is impossible to satisfy the Lambertian constraint. So far we have made the reasonable assumption that image noise is effectively zero relative to the surface orientation noise. We resolve situations where $I > a_S$ by assuming that the otherwise ignored noise of I becomes relevant. To be precise, as surface orientation is still far noisier than surface irradiance, we explain the data as much as possible with surface orientation by presuming the true irradiance, I_t, satisfies $I_t = a_S$ and then explain the difference between I_t and I by Gaussian noise in the image. The final per-pixel cost function is therefore

$$C_p(\hat{\mathbf{l}}, a_S) = \begin{cases} I < a_S & \varpi r + \varphi\sqrt{1 - r^2} \\ I > a_S & \varpi + \frac{(I - a_S)^2}{2\sigma^2} \end{cases} \tag{5.36}$$

using σ as the standard deviation of the irradiance noise.

5.3.3 Branch & Bound

This subsection continues the theme of simplification by removing albedo from the cost function; it steps into implementation however by specifying an optimisation procedure to do so. It is in fact core to the method, even though 'implementation' is delayed to the next section.

Albedo is constant over segments, this means that the per-segment cost now has to be considered. If S is the set of pixels in a segment then the per-segment cost is

$$C_S(\hat{\mathbf{l}}, a_S) = \sum_{p \in S} C_p(\hat{\mathbf{l}}, a_S) \tag{5.37}$$

which is simply the sum of the costs of all of the pixels in the segment. As elicited in the previous subsection, the per-pixel cost has a single minima, equation 5.35. It is then a decreasing/increasing function within the regions either side of the

minima. Obviously this property does not hold for a sum of such functions, as the minima do not necessarily align, but it does suggest a simple method of finding the optimal albedo value, for which this observation is instrumental.

Branch & bound[138] is an optimisation method that utilises a tree based search. Given that we have a cost function to minimise, which takes a set of parameters (Continuous, discrete or both.), a tree may be constructed by iteratively subdividing the space of possible parameters. Key to the idea is the ability to find bounds on the minimum value of the cost function for nodes of this tree; this allows the search space to be pruned - there is no point searching into a node of the tree if its minimum possible cost value is greater than the best value found so far. The upper bound for the minimum value of any parameter subspace also serves to provide a best value so far, even if the precise parametrisation has not yet been localised. With this model the tree can be exhaustively searched to a given depth, ultimately providing a small range in which the optimal value exists (For discrete parameters this range can cover a single parametrisation; this is not the case for continuous parameters however.). During the search a worst possible cost is stored; all nodes that could never achieve it are pruned. Good heuristics for choosing the order in which to search nodes, and how to subdivide the parameter space into nodes in the first place, will reduce the worst possible cost quickly; given such heuristics most of the search space can be quickly pruned.

Applying the method of branch & bound to the problem at hand is relatively straight forward. The cost function to minimise is equation 5.37, over a single parameter, a_S. Constructing the tree is therefore a matter of chopping up the range of a_S. Given that the range for a given node is $a_S \in [a_b, a_t]$ then it can be subdivided into two smaller nodes, with ranges $[a_b, 0.5(a_b + a_t)]$ and $[0.5(a_b + a_t), a_t]$. For the root of the search tree the range is set to $[0.01, 1]$, as an albedo of zero would cause problems; this defines the search tree, which is searched to a given depth. Next bounds on the minimum cost of a range are required. Firstly, define the cost for a range as an extension of equation 5.37

$$C_S(\hat{\mathbf{l}}, [a_b, a_t]) = \min_{a_S} \left(\sum_{p \in S} C_p(\hat{\mathbf{l}}, a_S) \right) \tag{5.38}$$

where $a_S \in [a_b, a_t]$. For an upper bound we can simply sample any point in the range - the minimum cost then has to be at worst that good. Due to the method of calculating the lower bound it is computationally convenient to find the minimum costs at the two extrema of the range, from which the smaller value is selected.

$$C_S(\hat{\mathbf{l}}, [a_b, a_t]) \leq \min_{a \in \{a_b, a_t\}} \left(\sum_{p \in S} C_p(\hat{\mathbf{l}}, a) \right) \tag{5.39}$$

As the cost equation is a sum of equations we can calculate a lower bound on the cost by summing the lowest cost for each equation in the sum within the range. This is effectively a lower bound by assuming all the lowest points line up; as the lowest cost is more likely than not outside the range this actually happens quite regularly at the range extrema. In other words this tends to give a poor bound only in the region of the answer, where the search will be going deep anyway. The minimum cost of the per-pixel cost function within a range is easy to calculate given that we can obtain the location of its only extrema, a minimum, from equation 5.35. If the minima is in the range then it is the location of the minimum cost; if its above the range then the minimum cost happens at the highest value, a_t; if its below the range at the lowest value, a_b.

$$C_S(\hat{\mathbf{l}}, [a_b, a_t]) \geq \sum_{p \in S} \left\{ \begin{array}{ll} a_t < a_m & C_p(\hat{\mathbf{l}}, a_t) \\ a_b > a_m & C_p(\hat{\mathbf{l}}, a_b) \\ \texttt{else}: & C_p(\hat{\mathbf{l}}, a_m) \end{array} \right\} \tag{5.40}$$

where a_m is the albedo for the current pixel at which the cost function is at its minimum, calculated using equation 5.35. A final detail is the order in which to search nodes; a priority queue on the minimum cost lower bound is used.

Given the above procedure we can calculate $C_S(\hat{\mathbf{l}})$, the cost of a segment for a given light source direction. The final cost, $C_M(\hat{\mathbf{l}})$, is simply the sum of the costs for all of the segments in the image,

$$C_M(\hat{\mathbf{l}}) = \sum_{S \in M} \min(C_S(\hat{\mathbf{l}}), t_S) \tag{5.41}$$

where t_S is a per segment cost cap, which is the number of pixels in the segment multiplied by a parameter to the algorithm. A cost cap is used to limit the damage of segments with bad orientation estimates; it is applied on a per-segment basis as the smoothing terms of stereopsis mean that bad surface orientation estimates will tend to be grouped into the same segment. This sequence of simplifications has ultimately provided a cost function with a single parameter, the light source direction.

5.4 Implementation

The result of the previous section is that the cost of any given light source direction can be efficiently calculated. This suggests a simple procedure - sample a number of light source directions and select the one with the lowest cost. The sampling method and further details are the subject of this section.

Stereopsis has already been discussed in section 5.2. Whilst stereopsis takes two images as input only one image is used as input for the other components of the algorithm, such as the segmentation needed to select regions with potentially constant albedo. Mean shift[53] is typically the first choice for a segmentation algorithm. In this case however an over-segmentation is preferred, to divide objects up so good data is less likely to be corrupted by grouping it with bad data. For this reason a k-means segmentation algorithm is used[6]. Initialisation consists of dividing the image into a grid, with each grid cell being an initial segment. The algorithm then runs to convergence, calculating the segment means using both image coordinates and colour coordinates. A slight deviation from normal k-means is used - pixels only consider assignment to their current segment and the segments of adjacent pixels; this is to obtain a sensible runtime and improve segment cohesion.

Given segmentation and stereopsis, the later converted to Fisher distributions on surface orientation, the cost for any given light source direction can be calculated; one further detail is that the camera response is calibrated for, see section B.4. Additionally, not all segments are used. Variance is calculated for irradiance and surface orientation[7]. Segments whose variance is too low, meaning they provide no information due to constant colour or orientation are pruned. Segments that are very dark are also pruned, as noise would be too high. Pruning these segments will typically remove bad data; it also reduces the computation required.

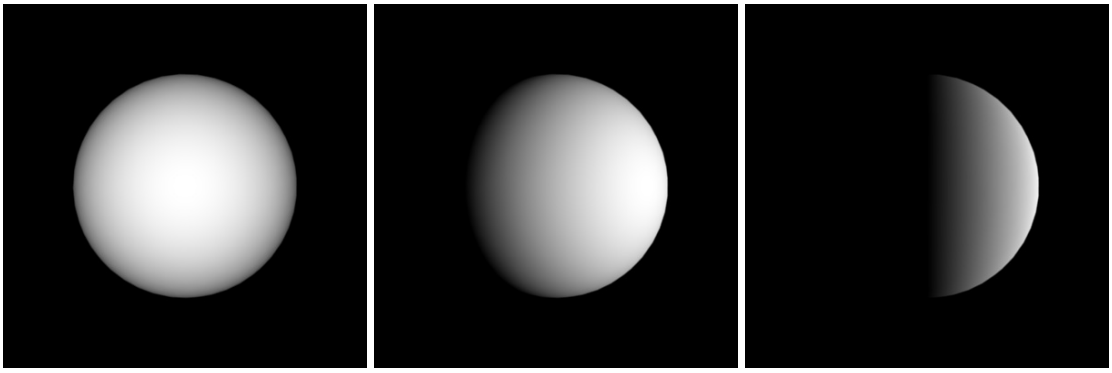
The final detail is the sampling procedure. Starting with a unit icosahedron it is subdivided a number of times, splitting triangles into four and re-projecting new

[6]Mean shift was tried, including with parameters to give an over-segmentation, but k-means gives better results.

[7]For surface orientation the angular nature is ignored - the unit direction vectors are treated as arbitrary 3D vectors. This is not a problem as most of the orientations for any given segment will be close together.

vertexes back onto the unit sphere. These vertexes then define directions from the origin to the candidate light source. Taking only the hemisphere of directions putting the light source behind the camera these directions are sampled. From this initialisation triangles sharing a vertex with the minimum in the sampling so far are recursively subdivided further, until the direction is sufficiently refined[8]. Using an icosahedron gives an approximately even sampling over the hemisphere, at least initially; refinement then takes a greedy approach by sampling around the current best value at higher resolution. As the cost function is quite smooth this procedure will typically find a good if not the best value, i.e. the frequency of the initial sampling is typically higher than the frequency of change for the cost function. The direction with the lowest cost is ultimately selected.

[8]The data structure allows triangles sharing an edge to have differing levels of subdivision.



(a) Sphere $0°$ (b) Sphere $45°$ (c) Sphere $90°$

Figure 5.3: Exemplars from the sphere data set. The degrees measure the light source rotation from being in-line with the camera. In the complete set there are also right camera images for the stereopsis - only left camera images are shown here. The complete set has an image for every $5°$ of light source deflection, from $0°$ to $90°$.

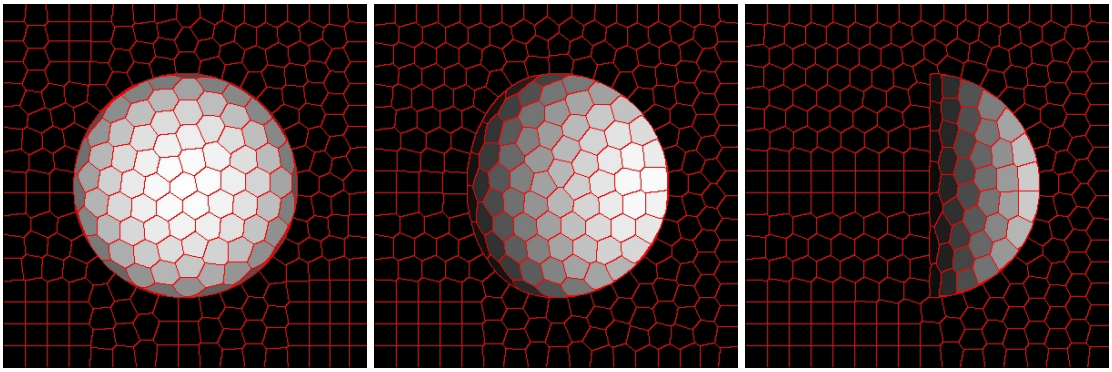
5.5 Experiments

Testing of the given algorithm is now presented. The field of light source estimation lacks anything in the way of a standard test set, or indeed even a review paper (Unsurprising given the variety of algorithms, where most are not comparable with many or indeed any others.). Additionally, the presented algorithm runs in a unique scenario, making its comparison to other algorithms effectively impossible. Testing is therefore limited to a quantitative evaluation, without any further comparison.

Both synthetic and real data sets are run; these form the following subsections. The synthetic results show that the algorithm works, however real world results are not brilliant - stereopsis is the issue, as the algorithm only works when given a good depth estimate. Fundamentally, the algorithm is limited by the requirement of applying stereopsis to a smoothly shaded input, which is unlikely to yield accurate results.

5.5.1 Synthetic

A data set consisting of a Lambertian sphere is presented first; it contains 19 images with the light source ranging from $(0, 0, 1)$ to $(1, 0, 0)$ in $5°$ increments, i.e. the directions $(\sin(\theta), 0, \cos(\theta))$ for $\theta \in \{0°, 5°, 10°, \ldots, 90°\}$. The images are



(a) Sphere $0°$ (b) Sphere $45°$ (c) Sphere $90°$

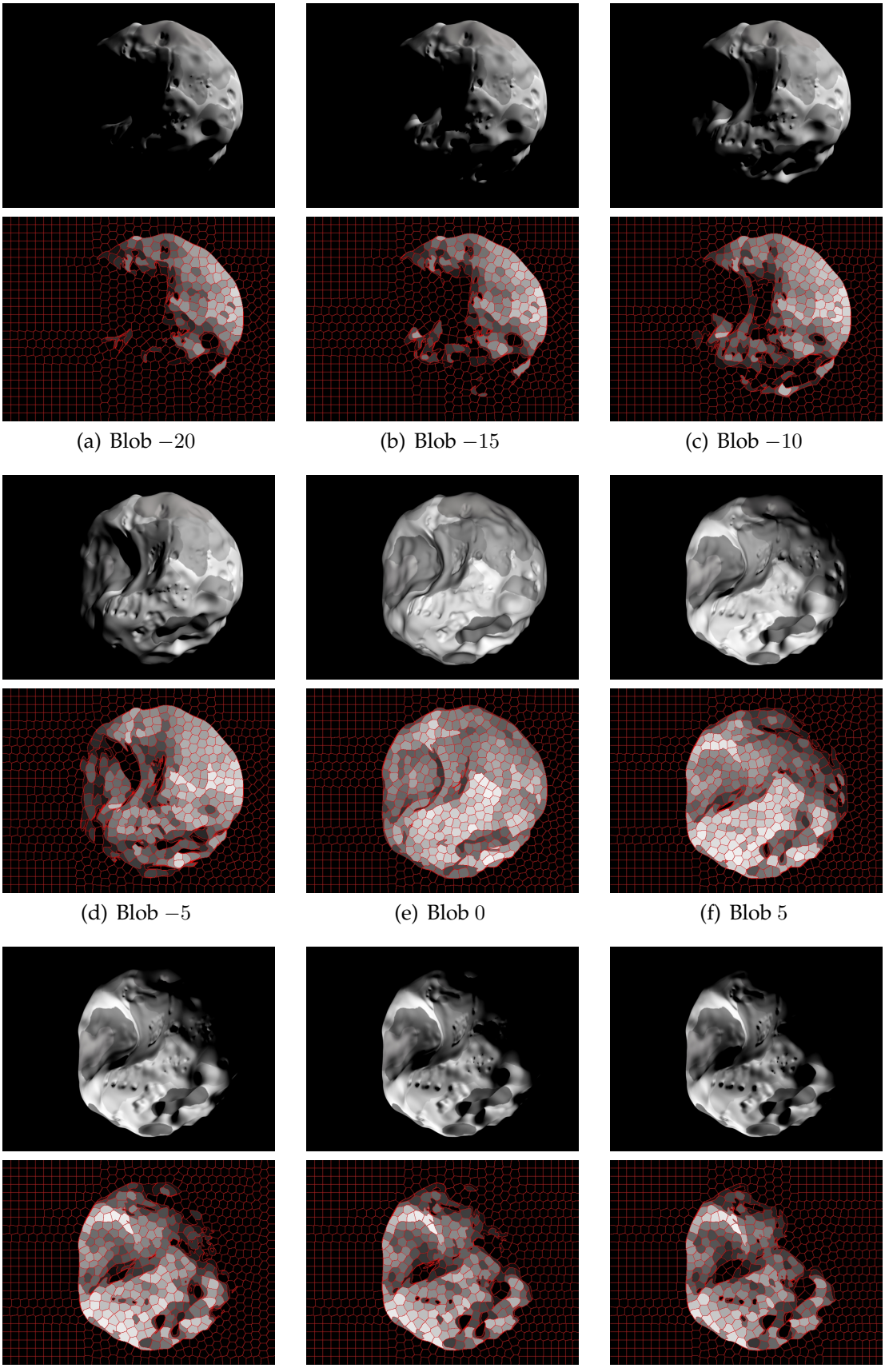
Figure 5.4: Example segmentations for the sphere data set, matching up with figure 5.3.

$0°$	$5°$	$10°$	$15°$	$20°$	$25°$	$30°$	$35°$	$40°$	$45°$	$50°$	$55°$	$60°$	$65°$	$70°$	$75°$	$80°$	$85°$	$90°$
$5.7°$	$6.4°$	$3.3°$	$1.7°$	$5.0°$	$1.8°$	$3.0°$	$11.6°$	$14.5°$	$19.4°$	$16.7°$	$22.6°$	$34.9°$	$41.8°$	$44.8°$	$43.1°$	$39.8°$	$45.5°$	$47.9°$

Figure 5.5: Results for the sphere data set. Light source angular deviation is given in the first row; angular error of the estimate is given in the second, both numerically and graphically.

labelled in terms of their deflection from $(0, 0, 1)$, which is the direction from the sphere to the camera. Figure 5.3 gives example images whilst figure 5.4 gives segmentations for the example images. Results are given in figure 5.5.

It is evident that as the light source moves away from the camera the error increases. When the light is within $30°$ of the camera the error is reasonable, and sufficient for SfS, however, as the deflection increases the error increases quickly, and soon provides unusable answers. This is not surprising as increasing light source deflection increases the number of shadowed pixels, which reduces the amount of available information. Additionally, the smoothing prior of the stereopsis algorithm is going to squash an untextured sphere, more so at the projected edge where it will cause larger surface orientation error. For larger light source deflections the spheres projected edge matters more, as it is the only illuminated area - this again contributes to greater error. A single albedo sphere is hardly ideal input due to this squashing, which primarily happens because the stereopsis algorithm struggles with a lack of texture. Fortunately, as will be demonstrated next, variable albedo improves stereopsis, leading to better results.



(a) Blob -20 (b) Blob -15 (c) Blob -10

(d) Blob -5 (e) Blob 0 (f) Blob 5

(g) Blob 10 (h) Blob 15 (i) Blob 20

Figure 5.6: Synthetic blob data set - each stack of two has the left image on top and the segmentation on the bottom; see text for details.

−20	−19	−18	−17	−16	−15	−14	−13	−12	−11	−10
70.5°	69.6°	68.6°	67.4°	66.2°	64.8°	63.2°	61.5°	59.5°	57.3°	54.7°
19.2°	9.2°	0.6°	4.3°	4.3°	11.4°	3.4°	6.0°	19.6°	3.2°	9.6°
−10	−9	−8	−7	−6	−5	−4	−3	−2	−1	0
54.7°	51.8°	48.5°	44.7°	40.3°	35.3°	29.5°	23.0°	15.8°	8.0°	0.0°
9.6°	8.3°	3.9°	8.8°	5.0°	9.2°	5.3°	4.9°	3.2°	5.7°	8.1°
0	1	2	3	4	5	6	7	8	9	10
0.0°	8.0°	15.8°	23.0°	29.5°	35.3°	40.3°	44.7°	48.5°	51.8°	54.7°
8.1°	8.0°	8.5°	4.9°	4.5°	1.6°	1.3°	4.2°	3.6°	1.8°	27.2°
10	11	12	13	14	15	16	17	18	19	20
54.7°	57.3°	59.5°	61.5°	63.2°	64.8°	66.2°	67.4°	68.6°	69.6°	70.5°
27.2°	3.8°	2.5°	6.7°	8.4°	5.2°	77.0°	80.5°	14.4°	15.5°	85.1°

Figure 5.7: Results for the blob data set. The results are split over four rows due to page width restrictions; each row overlaps by a single result with the rows above and below. Within each major row each result gets a column, containing three minor rows. The first is the light source direction number, the second the deflection of this light source direction from $(0, 0, 1)$; the remaining tall row contains the algorithms angular error, both numerically and graphically.

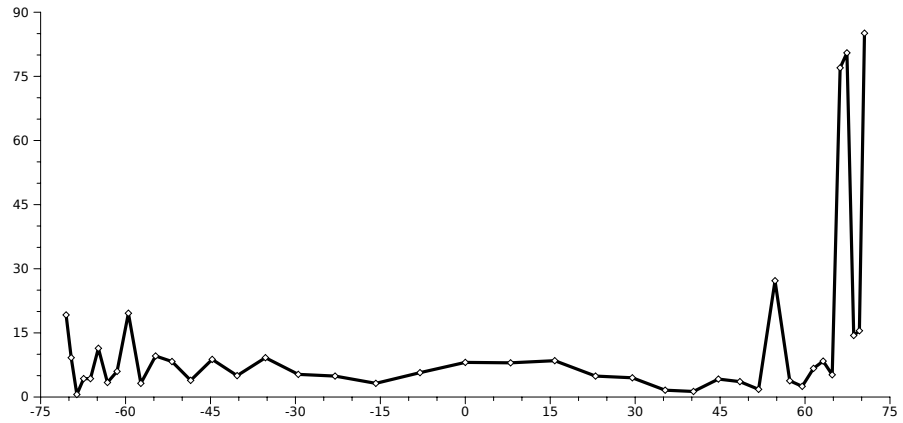


Figure 5.8: Graph of angular deflection (x axis) against angular error (y axis) for the blob data set. Note that results for negative light source direction numbers have had their deflection negated.

A highly deformed sphere with regions of constant albedo is now considered. Figure 5.6 contains a selection of images, and their segmentations, from this data set. The un-normalised light source directions for these images are given by $(n, n, 10)$ where $n \in \{-20, -19, -18, \ldots, 20\}$; the total number of inputs is therefore 41. This gives a denser sampling at higher deflections, which is the interesting region where the algorithm fails. Because the shape is not symmetric, unlike the sphere, the sweep of light source directions is done both ways to get more results, this time diagonally.

Results are given in figure 5.7, with a summarising graph also provided in figure 5.8. Unlike the sphere, with its gradual failure, this tends to either get it approximately right or fail, as indicated by the spikes in the graph. The error rate appears worse on the positive side of the light source direction sweep (The large spikes on the right of figure 5.8.); this can be attributed to stereopsis having problems with the shape of the bottom left of the image, probably due to the steep sides of the left side cavity. An examination of the segmentation shows that it is successfully separating the object from the background and mostly dividing it where albedo changes; when it is not successful the small segment size minimises the damage done. Overall this input shows that the algorithm can work when stereopsis produces a reasonable shape estimate, a task that is greatly assisted by the complex shape and variable albedo of this input.

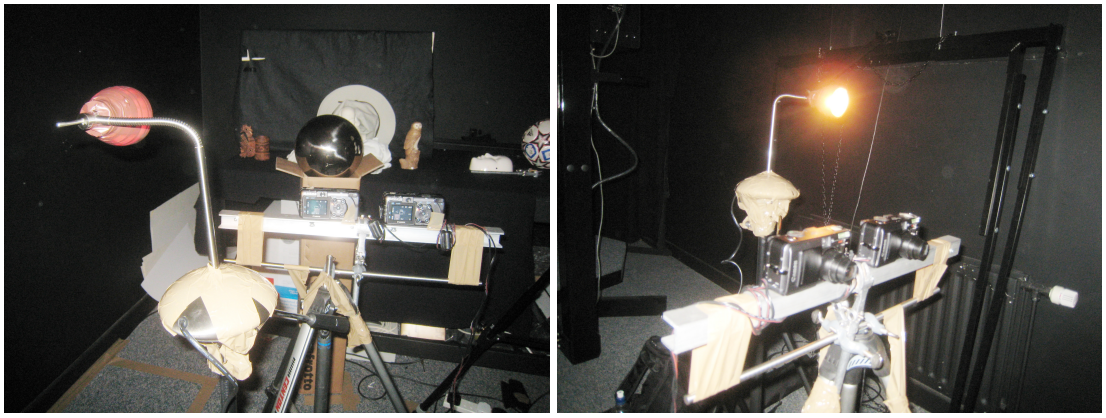


Figure 5.9: Two photos of the rig used to capture real world input. The light source has a bendable head and is taped to a tripod for easy positioning; it also has a small bulb with its diffuser taped over to make it better approximate a point light source. The separation between the camera lenses is approximately 16cm, with the scene around 1m away.

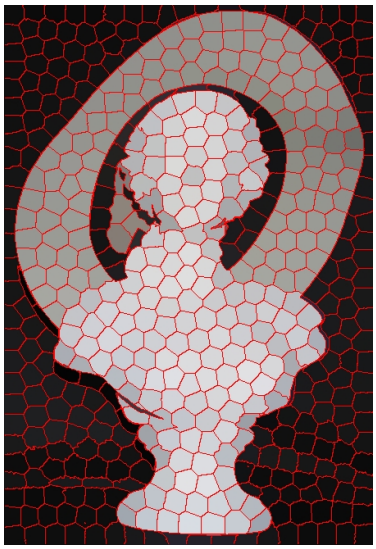
5.5.2 Real

Figure 5.9 gives the setup used to capture a data set consisting of 25 inputs. A single input is given in figure 5.10, this same scene with different light source directions is used for inputs 'a' to 'i'. Two further scenes are used, given in figure 5.11, corresponding to inputs 'j' to 'r' and inputs 's' to 'y'. The setup consists of a stereo rig in a dark room in front of a scene; a desktop light with an adjustable stalk and attached to a tripod is used so stereo pairs may be shot with the light source in a different position for each pair. To closer approximate a point light source the light sources diffuser cone is taped over; the light source being chosen for having a small halogen bulb, all the better to approximate a point light source with. Two Canon S70's are used, for which the response curve and intrinsic calibration are known. The fundamental matrix is calibrated for every input, from which a full geometric calibration of the cameras is determined; doing this for every input guards against the stereo bar being knocked between captures. Ground truth light source direction is determined using a reflective sphere, as shown in figure 5.10(d) - the position of the sphere is calculated by the user marking pixels on the edge of the sphere; once calibrated the reflection of the light source is selected and the direction from the sphere to the light source calculated[9]. As the sphere is

[9]During this step the sphere image has its brightness adjusted, to bring out the edges and better localise the light source reflection.

(a) Left image

(b) Right image



(c) Left segmentation

(d) Calibration image

Figure 5.10: Data set 'a' from the real world test set. Scene of several suitable objects captured in a dark room using a stereo rig. First are given the left and right images. Following is the left image segmentation and the calibration image. The calibration image is not rectified, unlike the rest. You can also see textured objects either side of the scene, these make geometrically calibrating the cameras automatically reliable.

(a) Left image for set ‘j’

(b) Left image for set ’s’

Figure 5.11: Sample left images from the other two scenes, complimenting 5.10(a) to provide a shot from each of the test scenes.

a	b	c	d	e	f	g	h	i
$5.1°$	$10.3°$	$13.8°$	$19.3°$	$15.4°$	$29.3°$	$35.4°$	$43.4°$	$34.7°$
$22.5°$	$3.4°$	$0.6°$	$6.5°$	$6.4°$	$9.1°$	$10.8°$	$7.9°$	$11.7°$
0.033 0.295 0.955	0.133 -0.082 0.988	0.234 0.021 0.972	0.433 0.021 0.901	0.266 -0.041 0.963	0.367 -0.062 0.928	0.522 0.118 0.845	0.599 0.020 0.801	0.495 0.060 0.867
0.001 -0.092 0.996	0.106 -0.146 0.984	0.240 -0.001 0.971	0.331 -0.015 0.944	0.224 -0.142 0.964	0.456 -0.175 0.872	0.576 -0.059 0.815	0.685 -0.067 0.726	0.554 -0.130 0.822

Figure 5.12: Results for the real world data set, inputs 'a'-'i', these correspond to the scene given in 5.10(a). Each column corresponds to a result; the first row is the input name, the second is the ground truth deflection of the light source from the direction to the camera, $(0, 0, 1)$. Row three gives the angular error, graphically and numerically. The remaining two rows are the estimated light source direction and ground truth light source direction, in that order. These two rows contain normalised column vectors.

positioned just in front of the scene this is a reasonable estimate of the direction to the light source, though it should be noted that the light source flares heavily in the reflection, so localisation is accurate to within a few degrees only.

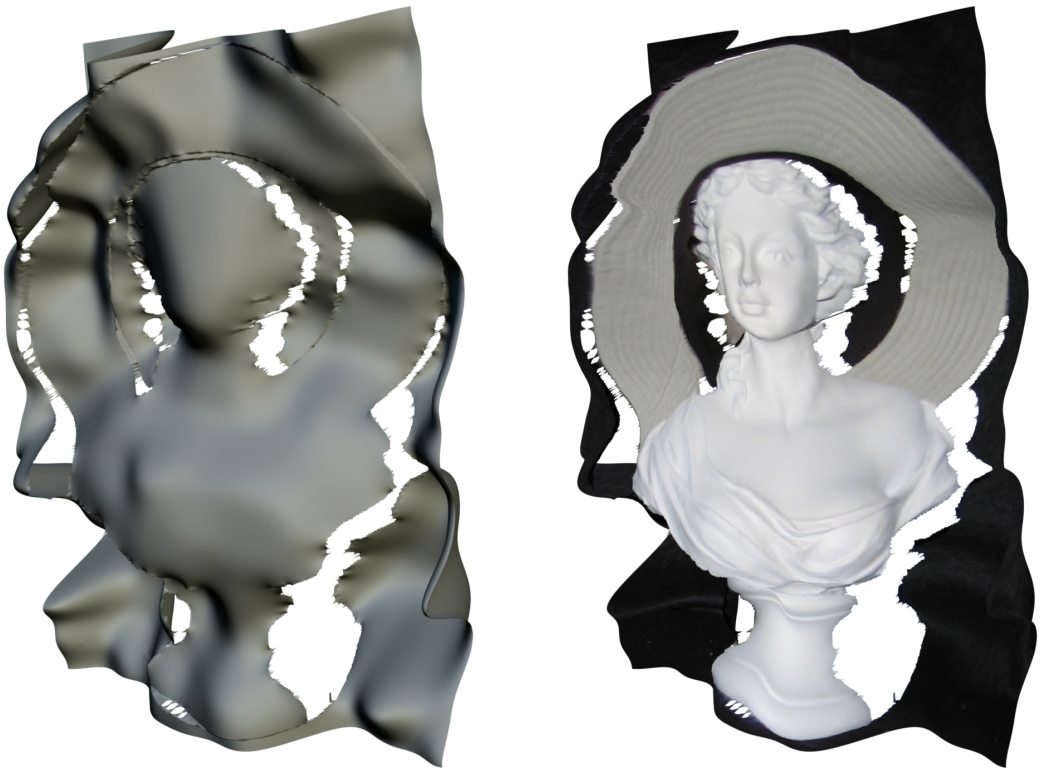
Results are given per scene, in figures 5.12, 5.13 and 5.14. The first scene, inputs 'a' to 'i', gives reasonable results, with the exception of a single outlier, input 'a'; it manages to do quite well past $30°$ of deflection. For the second and third scenes the results are worse, with the final scene, 's' to 'y', having effectively failed to produce results of the standard required for shape-from-shading. Figures 5.15, 5.16 and 5.17 provide renders of the stereopsis results for the input scenes. These renders effectively communicate the cause - the results are getting worse as the stereopsis loses accuracy. This correlation is of course no surprise, but it does highlight the scale dependency of the algorithm. In this particular case the details of the later scenes are at a higher resolution than the stereopsis output - the first scene does best simply because the input has a large smooth surface, whilst the later scenes suffer from curves at a detail level higher than the stereopsis can reasonably estimate. In addition to this whilst the segmentation method has been selected to limit the consequences of bad input it does include a scale dependency via its grid initialisation. For instance this is an issue with inputs 'j' to 'r' - the t-shirts

j	k	l	m	n	o	p	q	r
8.1°	14.3°	15.8°	13.8°	6.3°	9.6°	19.6°	16.9°	32.0°
15.1°	9.2°	1.4°	15.4°	9.9°	16.6°	21.9°	16.1°	12.0°
0.199 0.123 0.972	0.199 0.157 0.967	0.266 −0.082 0.961	0.200 0.041 0.979	0.167 −0.062 0.984	0.033 0.124 0.992	0.200 0.082 0.976	0.199 0.123 0.972	0.401 0.000 0.916
0.085 −0.110 0.990	0.247 0.007 0.969	0.256 −0.094 0.962	0.111 −0.213 0.971	0.000 −0.111 0.994	0.018 −0.166 0.986	0.162 −0.294 0.942	0.248 −0.151 0.957	0.502 −0.171 0.848

Figure 5.13: Results for the real world data set, inputs 'a'-'r', these correspond to the scene given in 5.11(a). See figure 5.12 for explanation.

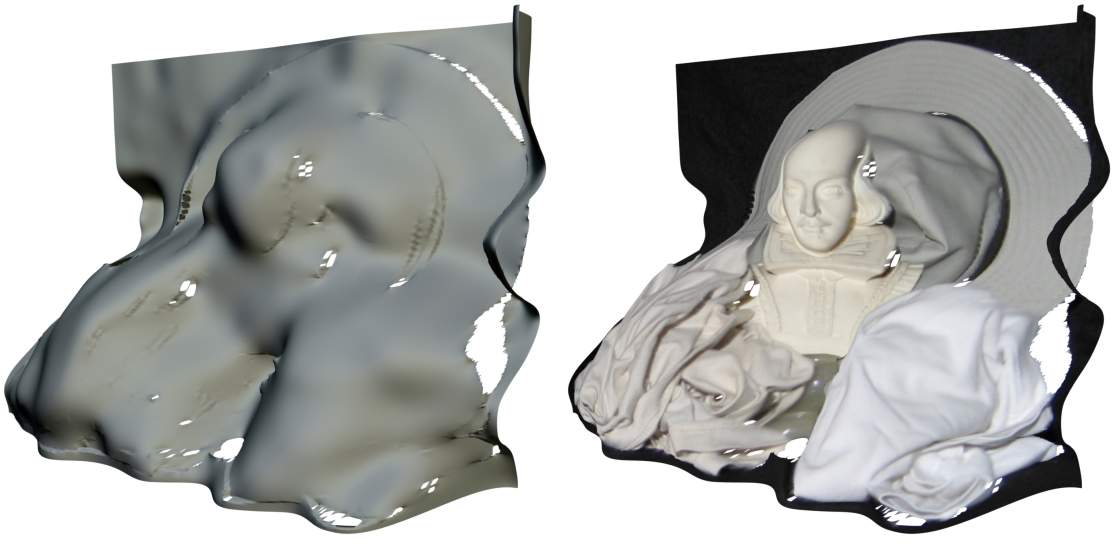
s	t	u	v	w	x	y
5.1°	12.8°	15.8°	18.6°	16.3°	8.1°	15.8°
18.5°	26.7°	30.8°	35.9°	34.8°	26.4°	35.2°
0.133 0.220 0.966	0.232 0.102 0.967	0.133 0.290 0.948	0.099 0.370 0.924	0.264 0.198 0.944	0.133 0.290 0.948	0.232 0.282 0.931
0.030 −0.082 0.996	−0.139 −0.172 0.975	−0.264 −0.063 0.962	−0.303 −0.096 0.948	−0.141 −0.242 0.960	−0.008 −0.143 0.990	−0.011 −0.271 0.962

Figure 5.14: Results for the real world data set, inputs 's'-'y', these correspond to the scene given in 5.11(b). See figure 5.12 for explanation.



(a) Untextured (b) Textured

Figure 5.15: Renders of the stereopsis result for input ’a’



(a) Untextured (b) Textured

Figure 5.16: Renders of the stereopsis result for input ’j’

(a) Untextured (b) Textured

Figure 5.17: Renders of the stereopsis result for input 's'

would be better with larger segments, whilst the bust of Shakespeare would be better with smaller segments. Whilst a new segmentation algorithm is probably not required more work is needed to adapt a current one to the requirements of this approach. The scale dependency of both stereopsis and segmentation is problematic, and whilst segmentation is principally solvable stereopsis is probably not. Detection of such scenarios might be possible however, so such segments can be ignored.

Ultimately stereopsis is the primary issue - for the algorithm to work you need objects that stereopsis can work with that also provide constant albedo regions with variable surface orientation; objects that satisfy this are quite rare. The scenes in question certainly provides lots of constant albedo surfaces, but stereopsis struggles due to the lack of texture; where texture is available its of little value as it only improves input regions that are of no use. These requirements are effectively at loggerheads, which is the problem with this approach in general.

5.6 Conclusions

A probabilistic algorithm for determining the light source direction from a stereo pair has been given. It is designed explicitly for the purpose of deriving a single infinitely distant light source, as required by a shape-from-shading algorithm. Starting from a simple probabilistic formulation the only major approximation made is to fit a Fisher distribution to stereopsis derived surface orientation; this is of limited relevance as it is a relatively simple modification to use a mixture of Fisher distributions for each pixel. The issue in doing so would be in deriving such a mixture from the stereopsis results - this might be reasonable with the current approach if using a stereopsis algorithm that provides several disparity estimates for some pixels.

Results are not spectacular, but given the limitations of the input this is to be expected. It is almost entirely an issue of stereopsis failing to provide sufficiently good results. This is evident in the error increase between synthetic and real world input, a relation which exists for stereopsis, and the evident worsening of the stereopsis result for the three real world scenes tallying with the error increases. A key issue is scale, as both segmentation and stereopsis are dependent on the scale of the image - for robustness segmentation has approximately equal segment sizes whilst stereopsis smooths away detail that is too fine for it to reliably estimate. These are areas to consider for future improvements. Another idea to consider is that of bluring - textured regions are currently down weighted, but texture is often restricted to a very small range of albedo levels - if the lighting frequencies and the texture frequencies are sufficiently separated it should be possible to blur away the texture whilst keeping lighting information; more information is of course to the advantage of the algorithm.

Further work to also consider is the current omission of a smoothing term between adjacent surface orientations in the original cost function; such a term could dramatically improve this algorithm. Of course, with such a term it would become a combined shape from shading and light source estimation algorithm, which is a potentially much more robust concept than running either alone. It would however be a very different algorithm, as none of the optimisations used in

this presentation would continue to apply. Another area of future work, though more a matter of curiosity, would be to compare this algorithm with the results of a human being - this would provide a baseline for evaluating the results. Ultimately an elegant working algorithm has been presented, but results are limited by the inputs. The most likely route to improve this algorithm is to tightly integrate it with an algorithm similar to the one of the previous chapter, and/or improve the ability of stereopsis to handle smoothly shaded input. A circularity, and irony, exists in this idea, as the algorithm has been developed to provide input to just such an algorithm. This suggests an iterative approach.

Chapter 6
Conclusion

ALL the work of the previous chapters is now summarised. This conclusion is split into two sections - the first summarises the work, focusing in on its novel aspects. The second section highlights the weaknesses of the approaches within, and the corresponding future work. It should be noted that this chapter intends to be an executive summary of previous conclusions, though it does consider further work in terms of the thesis as a whole.

6.1 Contributions

In chapter 3 a new algorithm for combining stereopsis and SfS is proposed. A modular approach is taken where the result are integrated[1] using Gaussian belief propagation. The results demonstrate success in areas where one algorithm will fail but the other can work, they also shows fine detail being provided by SfS and global detail being provided by stereopsis. In aesthetic terms the output is much prefered to stereopsis alone due to the smoothing and, more importantly, SfS provided fine detail. As part of this method a technique to assign standard deviations to disparity estimates is suggested; it later finds use for the light source estimation work. Also, the integration method is adapted to become an integration algorithm for visualising the SfS work by simply excluding the stereopsis information.

Motivated by poor results and a lack of probabilistic coupling the following research, chapter 4, proposes a new SfS algorithm. Again, belief propagation is used, this time with directional distributions. It is the first time the FB_8 distribution

[1] In both senses of the word - SfS is integrated in the calculus sense, whilst both techniques are combined together, i.e. integrated.

has been used within belief propagation - this is a particularly flexible directional distribution with the potential to be applied to other problems. A method to pass messages is devised for when the cost is the cosine of the angular difference between linked nodes; this is equivalently a method to convolve a FB_8 distribution by a Fisher distribution. The resulting algorithm allows messages to be passed analytically, albeit with approximation, but this makes for an extremely fast algorithm; due to the use of belief propagation it is also embarrassingly parallel, meaning a real time implementation is within reach. Results are better than all previous work except for Potetz[26], but then Potetz is several orders of magnitude slower and has only ever been presented for synthetic input. This difference is primarily due to the lack of an integration constraint in the presented algorithm.

Finally a light source estimation algorithm is presented in chapter 5, designed explicitly to provide the light source direction to a SfS algorithm. Uniqueness comes from the use of a stereo pair as input - the same input as to the first work of this thesis. Branch & bound is used to solve a relatively simple probabilistic formulation; the solution method contains little approximation. Surface orientation information, as provided by stereopsis, is approximated using a Fisher distribution, but this is not a major issue as a mixture could be used. A method of fitting a Fisher distribution to a disparity map is provided however - fitting a mixture could prove problematic. Given the limited information provided to the algorithm its output is of limited accuracy, it does however show reasonable results, which get worse as the usable input gets worse, implying the input is a limiting factor.

6.2 Weaknesses & future work

Most of the issues of the first work have, theoretically, been tackled by the later chapters. This also extends to the SfS work, where the issue with oblique light source directions would not exist on integration with the first work due to no longer needing gradient or boundary terms. Unfortunately the actual integration of these various systems has not in fact been done, so it is unknown if this is actually the case. This is hence the most valuable item of future work. It is

primarily an integration exercise, but involves work massaging input for the SfS step, and then extracting the results afterwards.

A fundamental assumption throughout this entire work is Lambertian reflectance - this causes issues, for instance when using human heads as input. Two possible solutions exist - updating the algorithms to use more sophisticated lighting models, or processing the input to make it comply with the model, i.e. removing non-Lambertian effects. Updating the algorithms is a lot of work, and it is questionable if it can be done without making them considerably more complicated, as the simplicity of the Lambertian model has been used to simplify repeatedly. There is also a question about infering the parameters of a more sophisticated model. Creating modules to manipulate the input to the Lambertian assumption seems more reasonable. Specularity removal is a well researched area, though there is the potential to improve such algorithms using the depth information made available by stereopsis, or the rest of the system iteratively. This does not handle other non-Lambertian affects however - rim lighting and sub-surface scattering [SSS] for instance. Creating a module to remove rim lighting is plausible, but removing SSS to leave a Lambertian surface is probably a tall order.

A key issue throughout the work as a whole is the stereopsis algorithm used - it is rarely appropriate. Two limitations drive this statement - it guesses when it has insufficient information and is discrete, though this last point is circumvented somewhat via adaptive smoothing. It can be observed that the integration algorithm is effectively a continuous stereopsis algorithm which also takes disparity differences as input, rather than simply smoothing under the fronto-parallel plane assumption. It however makes the crippling Gaussian assumption, hence the need for a separate stereopsis algorithm to select reasonable locations to approximate as such. If the Gaussian distribution were replaced with a distribution capable of representing the correlation information the integration algorithm would in fact be a stereopsis algorithm, and a separate stereopsis algorithm would not be required. A hierarchical construction would also be required to reduce the search space and get a reasonable run time, plus a suitable distribution for the inter-pixel offsets would be needed to get reasonable occlusion handling. Such an algorithm

is not hard to imagine - a piecewise linear probability distribution would suitably express the correlation information, and can be used with max-sum belief propagation without significant problems. Such an algorithm would also be ideal for providing input to the light source estimation algorithm. Stereopsis as an area of research in general focuses on better correlation and optimisation methods; the smoothing term between pixels has not been the focus of much research outside of occlusion handling. This work has shown that a more sophisticated handling of this smoothing term, in this case using SfS, is advantageous, and should probably be the focus of much more work.

Appendix A
Camera Geometry

An understanding of camera geometry is important to both the single view SfS problem and the two view stereopsis problem, and loses none of its importance when combining both techniques. A lot of the preceding content is covered in "Multiple View Geometry" by Hartley & Zisserman[139], though they are lacking most specifically in the area of rectification, which is important to the Stereopsis problem.

A.1 Single View Geometry - the SfS problem

SfS takes a single image as input, and is hence only concerned with the geometry of a single camera. Most SfS algorithms actually presume an orthographic projection, as demonstrated in figure A.1. For this model all pixels represent a cumulative ray of light in a single direction, but the locations in 3D space vary, usually in a grid pattern. This makes depth irrelevant, and all pixels represent an identically sized surface patch, once orientation has been factored out. The heavy use of this model can be put down to these points, as they simplify the mathematics greatly. Using this only the real world size of each pixel can be geometrically calibrated for, but that is usually irrelevant to SfS.

Real cameras use perspective projection[1], as acknowledged by some SfS algorithms (See 2.1.2.6). A perspective projection camera is modelled as a pin hole

[1] An orthographic camera is pragmatically speaking impossible in reality, though photocopiers and scanners are an exception, but only because they remove depth from the imaging process entirely. There are cases where an orthographic model makes sense however - specifically when a perspective camera is sufficiently far away from the imaged scene that the rays are close enough to parallel. Telescopes and spy satellites are the obvious examples, but any case where a strong zoom lens can be used is a possibility. There are also more esoteric cameras, ignored in this review, such as linear cameras[139, p.174].

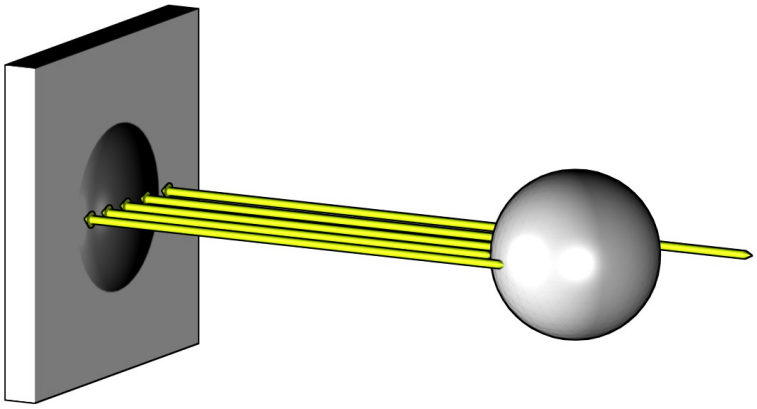


Figure A.1: A pictorial representation of an orthographic camera. On the left we have an imaging plane, on the right the imaged object. A slice of rays is indicated by the yellow arrows - the key feature is that they are all parallel to each other. This diagram is a $3D$ model rendered with orthographic projection.

camera, represented by figure A.2. In this case we can consider each pixel to represent light going to a single location - the pin hole, where the variable is the direction of travel. We now present a 2D pin hole camera in figure A.3 - we have presumed the camera is looking down the Z axis and that coordinates are all using the same unit - this is referred to as the *camera coordinate system*. For this diagram we have moved the focal plane in front of the pin hole - this is not a physically realistic representation, but it is mathematically identical and removes the mirroring of the image from consideration. Using similar triangles it is easy to see that a pin hole camera takes a point in space at (x, y, z), in the camera coordinate system, to the point $(fx/z, fy/z)$ on the image plane. Using matrices and homogeneous coordinates for the image coordinate but not the camera space position this may be represented as

$$\begin{pmatrix} fx \\ fy \\ z \end{pmatrix} = \begin{bmatrix} f & & \\ & f & \\ & & 1 \end{bmatrix} \begin{pmatrix} x \\ y \\ z \end{pmatrix} \tag{A.1}$$

We refer to the above matrix as the *intrinsic* matrix of a perspective camera, as opposed to the extrinsic matrix. The intrinsic matrix, $\mathbf{K}$, encodes the parameters of a camera that are constant, whilst the extrinsic matrix encodes the cameras position and orientation - the transformation from world coordinates to the camera coordinate system. Continuing with the intrinsic matrix, it is not yet general

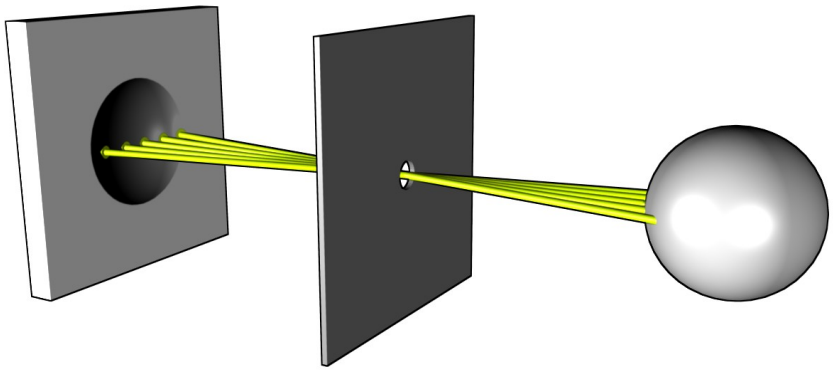


Figure A.2: A pictorial representation of a pin hole camera. It shows the imaging plane on the left and the object on the right, with a pin-hole in the centre. A slice of rays is indicated by yellow arrows - the pin hole only allows rays going through a particular location in space and so an image can form. This diagram is a $3D$ model rendered with perspective projection.

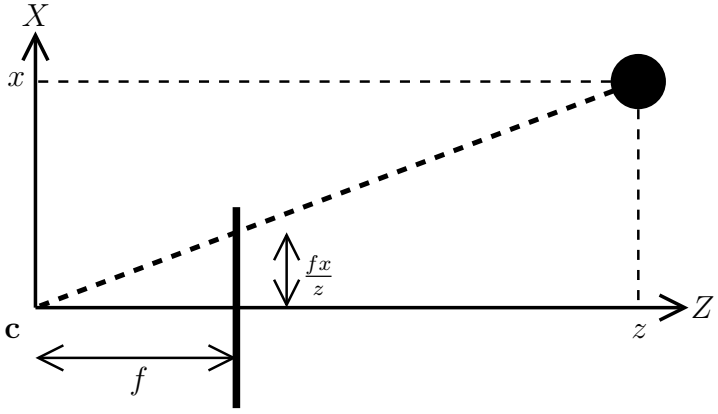


Figure A.3: A diagrammatic pin hole camera, given in $2D$. See text for details.

enough. The issue is that the coordinate system of the image can include an arbitrary scale and that the centre of projection will probably not be indexed as $(0, 0)$ on the image. To resolve these we introduce

$$\mathbf{K} = \begin{bmatrix} a_x & s & p_x \\ & a_y & p_y \\ & & 1 \end{bmatrix} \tag{A.2}$$

Here a_x and a_y are the focal lengths multiplied by an image scaling factor for the relevant directions; p_x and p_y define the principal point - the centre of the projection mapped to the image. This allows this matrix to output coordinates

scaled and offset to be in the images coordinate system. We also introduce s, a skew term, which can correct for the pixels coordinate system not being entirely perpendicular; for modern CCD based cameras this is invariably tiny if not zero however.

We now consider the extrinsic matrix. It moves from world coordinates to camera coordinates, so we may then apply $\mathbf{K}$ to get final image coordinates. Homogeneous coordinates are now used for the position in 3D space to obtain

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} = \mathbf{KR}\left[\mathbf{I}|-\mathbf{c}\right] \begin{pmatrix} x_w \\ y_w \\ z_w \\ 1 \end{pmatrix} \tag{A.3}$$

where the subscript i indicates image coordinates and the subscript w indicates world coordinates. $\mathbf{K}$ is the intrinsic matrix whilst $\mathbf{R}\left[\mathbf{I}|-\mathbf{c}\right]$ constitutes the extrinsic matrix. This matrix consists of two steps, the first, $[\mathbf{I}|-\mathbf{c}]$, uses the cameras centre, $\mathbf{c}$, to offset the coordinate system so that the camera is at the origin; $\mathbf{R}$ then rotates everything so the camera is looking down the Z axis, completing the transformation to the camera coordinate system. We then apply the intrinsic matrix to get image coordinates. If we multiply all three of these matrices we get $\mathbf{P} = \mathbf{KR}\left[\mathbf{I}|-\mathbf{c}\right]$, the $3X4$ camera projection matrix that combines all these steps into a single matrix. It is possible to decompose this matrix back to its constituent parts[139, p.163]. With pairs of image coordinates and matching 3D coordinates, which could be obtained using a calibration object, each pair provides a linear constraint on the matrix $\mathbf{P}$ - given enough it is then possible to solve for the camera projection matrix[2]. There are also techniques to solve for the intrinsic matrix alone using $2D$ calibration targets from multiple positions[140], which have the advantage that they can be printed. Once the intrinsic matrix is known solving for the entire projection matrix can be done from a single image of a $2D$ target.

Given the above understanding of the mapping from a point in space to an image point for a perspective camera we return to SfS, and specifically what

[2]Homogeneity and some constraints on the form of the matrix make it a little more complex than a typical linear problem.

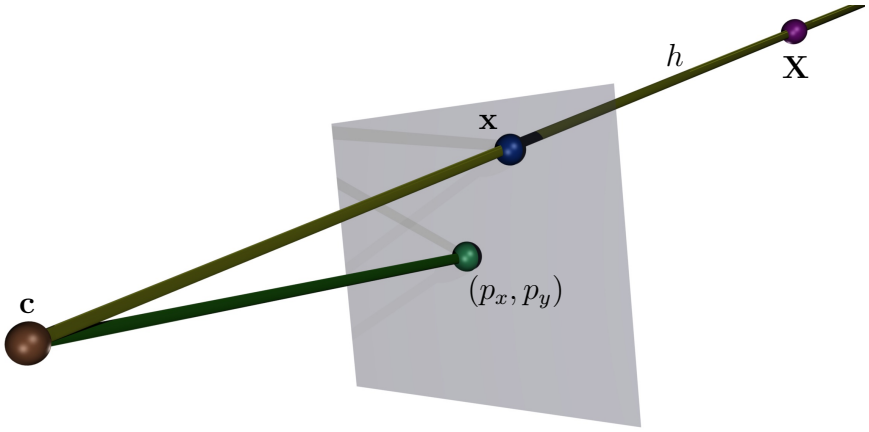


Figure A.4: Diagram showing the half ray associated with an imaged point, x. The camera centre, c (which projects to the principal point, p_x, p_y, on the grey imaging plane) and the imaged point define an infinite line. This line is necessarily restricted to a half-line (ray), h, by the requirement that the object, $\mathbf{X}$, be somewhere in *front* of the camera.

we know about the surface associated with a given pixel. Intuitively we have lost depth, which is to say the point could exist anywhere on a half-line (ray) and project to a specific point on the camera[139, p.161]. A half-line is defined by a location and a direction; the location is clearly the cameras centre, $\mathbf{c}$, the direction can be calculated by applying the pseudo inverse to $\mathbf{P}$ and applying it to a homogeneous image coordinate to get another point on the line; see figure A.4 for details. As this point can be on the line but behind the camera testing that it is in front of the principal plane is required to set direction. An alternative approach[139, p. 208] for direction is provided by the intrinsic calibration matrix. Given the intrinsic matrix K and a homogeneous image coordinate x then the un-normalised ray direction is $K^{-1}x$. This will then need normalisation and the application of the inverse rotation matrix.

We now have a fairly complete model of a pin hole camera, however in reality cameras need to use lenses in order to capture enough light in a reasonable time period. Lenses can distort the image, this can include colour distortions, such as vignetting[3] and achromatic distortion[4] but can also cause geometric distortion from

[3]Darkening at the edges.

[4]Different frequencies of light being magnified by different amounts and ending up miss-aligned.

the basic pinhole model. Whilst this distortion can be complex for a reasonable lens only radial distortion is worth considering. Radial distortion[139, p.189] can be considered a pre-processing distortion of the image before the pinhole model is used, where each pixel has its image coordinate adjusted by

$$\mathbf{x}_u = \mathbf{c} + L(|\mathbf{x}_d - \mathbf{c}|)(\mathbf{x}_d - \mathbf{c}) \tag{A.4}$$

$$L(r) = 1 + \sum_{n=1}^{N} f_n r^n \tag{A.5}$$

where $\mathbf{x}_u$ is the undistorted pixel and $\mathbf{x}_d$ the distorted pixel. c is the centre of the distortion, usually the principal point, and f_n are the distortion parameters. N gives the number of terms - it is usually set between 2 and 4 inclusive. The parameters can be estimated as part of the camera calibration, albeit using non-linear methods.

A.2 Two View Geometry - the Stereopsis problem

Two view geometry extends single view geometry to consider the relationship between two views of the same scene. This is the case with Stereopsis, where two view geometry is required to both prepare the data in a process called *rectification*, and then to post-process the results to get an actual $3D$ model, using *triangulation*. Both these steps are built on the instrumental concept of the *epipolar constraint*.

A.2.1 The Epipolar Constraint

Figure A.5 gives a diagrammatic representation of two cameras viewing a single point, $\mathbf{X}$. We have two cameras, their centres denoted by $\mathbf{c}$ and $\mathbf{c}'$ - the key observation is that these three locations in space - $\mathbf{X}$, $\mathbf{c}$ and $\mathbf{c}'$ - define a plane. This plane then intersects the imaging planes of the cameras, where it forms the epipolar lines, marked as $\mathbf{l}$ and $\mathbf{l}'$. Each epipolar line must then contain the projection of $\mathbf{X}$ onto the camera, marked as $\mathbf{x}$ and $\mathbf{x}'$. It must also contain the projection of the other cameras centre onto the current cameras imaging plane, these are marked as $\mathbf{e}$ and $\mathbf{e}'$ in figure A.5. These points are referred to as the

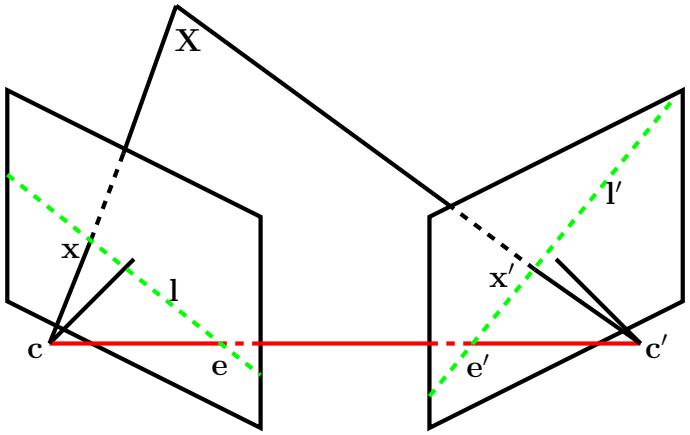


Figure A.5: The epipolar constraint. See text for details.

epipolar points and have the property that all epipolar lines must pass through them. Additionally, triangulation does not work at epipolar points, and becomes unstable to noise in the vicinity.

Consider a projected point onto the camera denoted by $\mathbf{c}$, so we know $\mathbf{x}$ but not $\mathbf{X}$. We know from single view geometry that this point restricts $\mathbf{X}$ to a line in space. If we knew $\mathbf{c}$, $\mathbf{c}'$ and some calibration information we could then construct the plane which would give us the epipolar line in the other image, $\mathbf{l}'$. We would then know to only search this $1D$ line for a match to the current point, which considerably reduces the search space of the correspondence problem solved by stereopsis.

Fortunately we can calculate the epipolar relationship without knowing large amount of information about the cameras. Specifically if we represent a line by $\mathbf{l} \in \Re^3$ such that

$$\forall \mathbf{x}; \mathbf{l} \cdot \mathbf{x} = 0 \implies \texttt{Point is on line} \tag{A.6}$$

then there is a $3X3$ homogeneous matrix $\mathbf{F}$ such that

$$\mathbf{l}' = \mathbf{F}\mathbf{x} \tag{A.7}$$

This is the *Fundamental matrix* and is unique for each camera combination. More importantly, it is a linear relationship where a matched pair of points constrains

one degree of freedom - this means that given seven matches[5] between two images you may calculate the fundamental matrix and restrict to $1D$ all further searching for matches. Algorithms, usually based on RANSAC[74] and a feature detector[73], exist to reliably calculate this from just an image pair[139, p.279].

A.2.2 Rectification

Rectification[139, p.302] is the well understood process by which an image is transformed so that the epipolar lines are mapped onto the scan lines. This removes the dependence on camera calibration from the problem, leaving only a 1D correspondence problem to be solved by the actual stereo implementation - it is almost invariably used to pre-process data before stereopsis. Whilst the fundamental matrix defines which lines of the two images must match up it does not overly constrict the solution, so multiple algorithms exist to select one.

Most algorithms are linear and consist of selecting a $2D$ homography for one or both images. Examples include Hartley & Zisserman[139, p.305] or Loop & Zhang[141]. Such a technique has to select a pair of projective image transforms that minimise image distortion. This is because an arbitrary but constraint satisfying transform could potentially scale one part of the image to be many times larger than another part. As most algorithms then scale the rectified image so no information is lost this could result in the image being orders of magnitude larger when it reaches the stereopsis algorithm.

Linear rectification algorithms have advantages, mostly in terms of simplicity to implement, and them being linear makes triangulation particularly easy, as covered in the next subsection. The major disadvantage is that rectification has to map the epipolar points to infinity - if these points are near the image the rectified image could be very large; if they are in the image the rectified image has to be infinitely large so a linear method fails. Epipolar points appearing in the image will happen, for instance, if the translation between the cameras is mostly forward or backwards, or if one camera can see another, and so is not particularly uncommon.

[5]Homogeneity reduces the 9 parameters to 8, the matrix is also required to be singular, which reduced the degrees of freedom to 7. In reality many more matches would be used for robustness.

Non-linear algorithms exist to avoid the above problems, most notably the epipole in the image issue. The most notable example is the polar algorithm of Pollefeys et al.[142], which is probably optimal. It considers the fact that epipolar lines are actually half lines, starting at the epipolar point, and then puts these half-lines in a polar coordinate system. Generation of the final image is then done such that pixels appear roughly the same size, so none of the scaling problems exist. It handles there being an epipole in the image by mapping it to an edge of the rectified image. Examples of both linear rectification and polar rectification are given in figure A.6.

A.2.3 Triangulation

Stereopsis will match pixels and in effect claim that the same surface point in $3D$ space projects back to the matched pixels. Triangulation is then required to obtain the $3D$ point. Given image coordinates in both images and camera projection matrices for both images the task is to find a $3D$ location that minimises the re-projection error[139, p.310]. If the points satisfy the epipolar constraint, which they should if they have come from a stereopsis algorithm, then a simple linear method is optimal[139, p.312], otherwise a more sophisticated method is required[139, p.315].

There is an issue with the above - the assumption that camera projection matrices are known, an assumption which is false as we only have the fundamental matrix. The fundamental matrix provides a constraint on the possible projection matrices[139, p.244]

$$\mathbf{F} = [\mathbf{e}']_X \mathbf{P}' \mathbf{P}^+ \tag{A.8}$$

where $[.]_X$ makes a skew symmetric matrix from a vector, and $.^+$ implies the pseudo-inverse. Position and rotation in space can not be fixed as we have no coordinate system, and so we arbitrarily set P to be at the origin looking down the Z axis. Given these constraints we have canonical projection matrices defined as[139, p.256]

$$\mathbf{P} = [I|\mathbf{0}] \qquad \mathbf{P}' = [[\mathbf{e}']_X \mathbf{F}|\mathbf{e}'] \tag{A.9}$$

(a) Linear A (b) Linear B

(c) Polar A

(d) Polar B

Figure A.6: Examples of rectified images - the inputs were all rectangular. Both a linear algorithm[141] and a polar algorithm[142] are demonstrated. The polar example is one where the epipole is in the image, and has have been rotated by 90° for convenience of display, so searching is along the y-axis, rather than the more typical x-axis.

where $\mathbf{e}'$ is an epipolar point, which is the right null space of $\mathbf{F}$. The projection matrices are obviously not correct - this is one solution from a family of matrices that match the given fundamental matrix. If triangulation is done with these matrices the resulting reconstruction will be correct to an arbitrary $3D$ homography, which can be a very strong distortion, so doing better would help. Additionally, the centre of $\mathbf{P}'$ is at infinity, which can be inconvenient. Intrinsic calibration is relatively easy to calibrate for offline, and can be used to get a theoretically correct estimate, ignoring the global transformation and rotation. In reality the error in the various estimation methods makes this a minimisation problem, where the resulting matrices would generally exactly match the fundamental matrix but be allowed to vary from the intrinsic matrices due to their relative accuracy.

The previous works for all coordinates, regardless of if they come from a rectified image or not - rectification can make this a simpler problem. In the case of linear rectification[6] the equation is simply[84]

$$z = \frac{bf}{d+n} \tag{A.10}$$

This relies on the concept of disparity, d - as we know the matched coordinates in two rectified images are on the same scanline and hence share a y coordinate we can define the two points as (x, y) and $(x + d, y)$. Stereopsis algorithms mostly output disparity. b is the baseline between the cameras and f is the focal length, and they have to use the same units as d. Additionally, the coordinate systems of the two images have to use the same origin; as this is invariably not the case n is introduced as a correcting offset.

[6]Non-linear rectification breaks equation A.10, however, for the polar rectification technique each scanline is handled in a linear method, so this method works with different parameters for each scanline rather than for each image.

Appendix B
Light

THE previous section considered the geometry of cameras - we now consider the light they detect. Whilst camera geometry was relevant to both stereopsis and SfS it mattered more to stereopsis - the reverse is true here, as this section is primarily concerned with the equations behind SfS, though still relevant to stereopsis. Below we discuss realistic light modelling, but in further subsections we simplify it to something we can computationally optimise with. We end this section by considering the relationship between actual light captured and the image provided by a camera.

B.1 The Bidirectional Reflectance Distribution Function

Consider a ray of light - it is emitted from a light source to travel through the air until it hits an object. When it hits an object the light is re-emitted, turning the object into a secondary, though much dimmer, light source. This process repeats, until eventually a light ray enters the camera and hits the CCD, where it registers as an electrical signal. The processing of captured light into an image by a camera is considered in section B.4. At each collision with an object the material of the object affects how the light is reflected. This is a probabilistic process, with single quanta of light being bounced through the object differently; in practise the numbers of quanta are so large that a continuous analysis is reasonable. We now introduce the Bidirectional Reflectance Distribution Function [BRDF][143]. The basic formulation takes two parameters - the incoming light direction and the out-

going light direction, and returns the ratio of differential radiance over differential irradiance[144, p.32]. Radiance is defined as the flux (Light power, measured in watts) per unit projected area per unit solid angle[144, p.20]; irradiance is just flux per unit surface area. Solid angle can be thought of as angular area, and is the 3D equivalent of the radian. One steradian, the measure of solid angle, is defined by the angular area needed on a sphere of radius r to create an area of r^2 on the sphere, much like a radian is defined as the angle on a circle of radius r to create an angle covering a distance r on the circle. These two directions are given relative to the surface orientation, so this function has 4 degrees of freedom. Obviously, unless the object is itself a light source, or some kind of light amplifier, the integral of this function over the outgoing direction for each incoming direction should not be greater than one. It may be less than one as some of the light may be absorbed and converted to an alternate form of energy, usually heat. Less obviously Helmholtz reciprocity applies, which states that if you swap the incoming and outgoing directions you should get the same result. One parameter arguably omitted from the above BRDF definition is location, as the BRDF will usually vary as you move across a surface, or indeed through it.

The original BRDF with two parameters needs further consideration when simulating reality. Firstly, the BRDF is often assumed to apply to solid materials only, and therefore only cover a hemisphere of angles - for transparent objects it has to consider an entire sphere however. Frequency is often added to the parametrisation, normally the frequency of light for both incoming and outgoing, though it is possible for a reflection to have a different frequency (fluorescence) and hence require a separate value for each; this is rarely modelled however. Often, instead of modelling frequency directly a parametric BRDF will simply assume that it is provided a different parameter set for each frequency (i.e. one for red, one for green and one for blue.). Omitted also from the BRDF is time - for instance objects that absorb light, store it chemically and later re-emit it (phosphorescence), such as glow in the dark stickers; also light can enter an object at one point and be emitted at another point, which is referred to as subsurface scattering - the human face exhibits such behaviour. Neither of these effects are normally simulated directly however, as they are too computationally demanding outside of simple

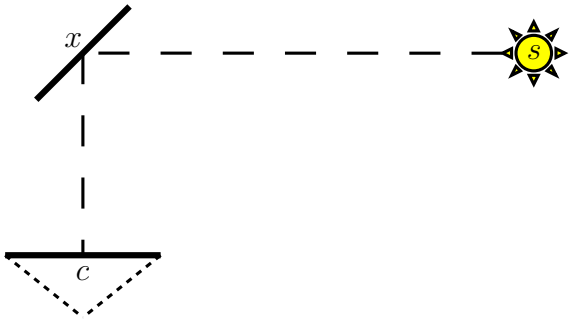


Figure B.1: A light ray travelling from a light source, s, to a point on a surface, x, and then being re-emitted and striking a camera, c. See text for further details.

test examples - various approximations are the norm if these effects are required for production rendering. Pragmatically a BRDF with its four degrees of freedom is generally not modelled as an arbitrary function, instead parametric BRDFs with a small parameter set that reasonably simulate certain types of material are used. These will be the subject of the following subsections.

B.2 Lambertian surfaces

A surface exhibits Lambertian reflectance if its perceived brightness does not depend on the angle of the viewer. In consequence its BRDF equation must be constant, giving

$$f_r(\hat{\mathbf{d}}_{in} \rightarrow \hat{\mathbf{d}}_{out}) = \frac{a}{\pi} \tag{B.1}$$

where $a \in [0, 1]$ is the ratio of reflected light from direction $\hat{\mathbf{d}}_{in}$ to direction $\hat{\mathbf{d}}_{out}$. Lambertian surfaces are also referred to as diffuse surfaces, having the property that they reflect light equally in all directions - they are the uniform distribution.

The above, equation B.1, may appear somewhat foreign - in computer graphics it is typically given in terms of the irradiance, I, that is actually received by the camera, under a set of specific assumptions. Consider figure B.1, showing a light ray travelling from a light source to a surface, where it is re-emitted in the direction of the camera. We start with the assumption that there is no participating medium, so light is not attenuated or scattered as it passes through space - this is effectively

assuming a vacuum, though is a reasonable assumption in air[1]. This means that the radiance leaving the light source, $L(l \rightarrow x)$, is the same as the radiance arriving at the point on the surface, $L(x \leftarrow l)$, i.e. $L(l \rightarrow x) = L(x \leftarrow l)$. The BRDF is defined as[144, p.32]

$$f_r(\hat{\mathbf{l}} \rightarrow \hat{\mathbf{v}}) = \frac{dL(x \rightarrow c)}{L(x \leftarrow l)\cos(\theta)d\hat{\mathbf{l}}} \tag{B.2}$$

where $\hat{\mathbf{l}}$ is the direction to the light source and $\hat{\mathbf{v}}$ the direction to the viewer, i.e. the incoming light direction and outgoing light direction respectively. The cosine term exists because the surface area struck by incoming photons depends on the angle the surface makes with the light source; θ is the angle between the surface normal, $\hat{\mathbf{n}}$, and the direction to the light source, $\hat{\mathbf{l}}$. This can be rearranged to give us the outgoing radiance for a specific direction,

$$L(x \rightarrow c) = \int_{\Omega_x} f_r(\hat{\mathbf{l}} \rightarrow \hat{\mathbf{v}})L(x \leftarrow l)\cos(\theta)d\hat{\mathbf{l}} \tag{B.3}$$

where Ω_x indicates the entire hemisphere of incoming directions to x. In words the output for a given direction, in this case from the surface point x to the camera c, is an integral over incoming light from every direction, where the BRDF decides how much energy is taken from each incoming direction. Note that the BRDF units are 'per solid angle'.

We now assume a point light source; this is an issue as a point light has no area. This is resolved by pretending it does, whilst still allowing it to come from an infinitely small location. Effectively the integration is now over a Dirac delta function, so we can simplify to

$$L(x \rightarrow c) = \pi f_r(\hat{\mathbf{l}} \rightarrow \hat{\mathbf{v}})L(x \leftarrow l)\cos(\theta) \tag{B.4}$$

where π is the solid angle of a hemisphere, required as $L(x \leftarrow l)$ is for a unit of solid angle and the integration was over a hemisphere of solid angle. The `distance`^{2} falloff of a light source has been ignored - in reality as an object moves away from a light source the solid angle it subtends reduces, so the light energy received per

[1] Assuming the temperature to be approximately constant - a temperature gradient results in diffraction and bends the light; if the gradient is large enough it can cause a mirage.

surface area of the receiving surface is reduced. Assuming an infinitely distant point light source, as is typical for SfS algorithms, is ultimately about assuming that there is no falloff, or more accurately that the light source is sufficiently far away for falloff to be inconsequential. Define $L(x \leftarrow l)$ as the length of the vector $\mathbf{l}$, $|\mathbf{l}| = L(x \leftarrow l)$, where the direction of $\mathbf{l}$ is $\hat{\mathbf{l}}$. Using this and also substituting in the Lambertian BDRF

$$L(x \rightarrow c) = \pi \frac{a}{\pi} |\mathbf{l}| \cos(\theta) \tag{B.5}$$

The final step is the observation that a camera receives radiance, but converts it into irradiance by projecting each angle to a unique spot on the sensor[2]; essentially we can just substitute $L(x \rightarrow c)$ with I[3]. It might appear that the distance between the sensor and the object have been ignored, as increasing the distance from the camera to the object means the camera subtends a smaller solid angle and receives less light. However, as the object moves away it fills in less space on the sensor, so the actual pixels remain equally bright, as these two affects precisely cancel out. This gives the Lambertian shading equation,

$$I = a\mathbf{l} \cdot \hat{\mathbf{n}} \tag{B.6}$$

where, to reiterate, I is the image irradiance, $\mathbf{l}$ as the light source direction multiplied by the light source strength, $\hat{\mathbf{n}}$ the surface normal and a the albedo. This makes use of the fact that the dot product of two vectors is the cosine of the angle between them, multiplied by the lengths of the vectors. One advantage of using a shading model is that a BRDF can only give a zero result at 90° to the light source, a shading model can do otherwise, and model rim lighting etc.[4].

Perfectly Lambertian surfaces are rare, however, most surfaces exhibit some Lambertian behaviour, and it is often the case that the majority of light reflected back by an object is following the Lambertian model, with details such as rim-

[2]Consequentially it is quite reasonable, and some authors do, to refer to the pixels as measuring radiance.

[3]This is also making use of the no participating medium assumption from earlier, so that $L(x \rightarrow c) = L(c \leftarrow x)$.

[4]A BRDF assumes that the surfaces surface geometry is perfectly known, it is more practical to assume roughness modelled in the shading function rather than the surface geometry representation.

lighting and specularities being a relatively small component in the response. Most of the more sophisticated shading models consist of a Lambertian term followed by further terms, to cover such model improvements. The Lambertian shading model is very simple, and also quite general, as it can handle a large percentage of real world objects - it is not surprising that it is the common model in SfS algorithms. Stereopsis algorithms almost invariably make a Lambertian assumption, specifically because they measure correspondence using image irradiance. If a surface is not Lambertian its brightness will change as the camera moves and correspondence based on measuring brightness differences will fail. This is evident as stereopsis algorithms fail with mirrors and in the regions of specularities.

B.3 Other Lighting models

In this thesis we use the Lambertian model exclusively, however, for completeness we now iterate some of the more advanced parametric models.

B.3.1 Torrence-Sparrow

The Torrence-Sparrow model[145] takes a typical approach - modelling the diffuse component with the Lambertian equation, B.6, and then adding in a specularity term. The specularity term is physically based, treating the surface as consisting of micro-facets with a distribution on their direction. The specular highlight is then calculated from the surface area of the facets that perfectly reflect the light to the viewer. Its shading equation is given by

$$I = a\mathbf{l} \cdot \hat{\mathbf{n}} + \frac{dgf}{\hat{\mathbf{n}} \cdot \hat{\mathbf{v}}} \tag{B.7}$$

where you can see the diffuse component on the left and the specular component on the right. $\hat{\mathbf{v}}$ is the vector pointing at the viewer - the divisor of the specular term exists to compensate for the foreshortening of looking at a surface obliquely, as indicated by Lambert's cosine rule. The three variables d, g and f then define the specular strength:

- d, distribution term, this is the distribution of micro-facets. The original paper uses a Gaussian distribution, so $D = \exp(-\alpha^2/\sigma^2)/\sigma\sqrt{2\pi}$. α is the angle between the surface normal and half-way vector, essentially a measure of how far we are from the perfect reflection direction. σ is the distributions standard deviation, a parameter which is set high for a tight specularity. Other authors have used alternate distributions, for instance Hara et al.[115, 116] used a Fisher distribution.

- g, geometry term, this compensates for micro-facets occluding each other, being the fraction of un-occluded facet. An approximation, unsurprisingly. Its inclusion matters as when included and taking limits on the potential divide by zero of the foreshortening term it no longer heads to infinity.

- f, Fresnel term, provides the percentage of light that is not absorbed and therefore reflected back as a specularity, as a function of the incident light direction and the index of refraction.

B.3.2 Ward

Ward[146] is quite similar to the Torrence-Sparrow model, but instead of a 1D Gaussian it has a 2D Gaussian[5]. It is an anisotropic distribution, which is to say the materials shading is directionally dependent, and requires the use of the surfaces tangent direction. Unsurprisingly, this leads to specularities with a non-circular shape, in this case an elliptical shape due to the 2D Gaussians covariance matrix being a parameter. An example of such a material would be brushed aluminium, as the brushing process breaks the isotropic assumption on micro-facet orientation and biases them towards a single direction.

B.3.3 Oren-Nayar

Both of the previous examples have a Lambertian term plus a specularity term, we now give an example of a model that ignores Lambert's cosine rule, the Oren-

[5]It is based on similar ideas to Torrence-Sparrow, so the terms multiplying the distribution have the same purpose, but different assumptions are made, and hence the terms have different formulations.

Nayar shading model[35][6]. It takes a micro-facet approach, but this time for the diffuse component rather than the specular component. This matters as if the surface is not precisely smooth then foreshortening will apply to the viewer, and the surface will get darker as the viewer direction moves away from the light source direction. A Gaussian is again used to simulate the distribution of micro-facets. Experimentally this model is shown to offer improved results for rough surfaces, such as sand and plaster.

B.4 Camera Response Function

The irradiance that lands on a cameras sensor does not go through a linear response on its way to the image. This concerns a SfS algorithm as without a linear response the assumed Lambertian reflectance will not apply in the provided image. Actual response is typically an S-shaped curve - this is to emphasis detail in the bright and dark areas, and is a good method of re-mapping a large dynamic range to a smaller dynamic range for human perception[7]. Fortunately this curve can be easily calibrated for by taking a sequence of shots of a white surface in constant lighting, where only the cameras exposure time is varied. A polynomial function can then be fitted as we know that the relationship between exposure time and irradiance must be linear. All data used in the following research has been calibrated as such, though typically we show the uncalibrated images as they are perceptually better to look at.

[6]This model does not include a specularity term - you would generally add such a term from another model for a complete shading function.

[7]For some recent Canon models of camera, such as the 5D Mark 2, you can edit the default curves (Which are all S-shaped.), and provide custom curves, including a straight line. Other camera manufacturers can be assumed to shortly follow suite.

Appendix C
Belief Propagation

BELIEF PROPAGATION [BP] is instrumental to this work, and is used extensively in chapters 3 & 4. It is a technique that has gained widespread use within the computer vision community, especially in the field of stereopsis where most of the top algorithms use it at the time of writing[47, 48]. SfS has also been tackled successfully with it[26], as have many other problems[79, 147, 38, 148]. Indeed, its success has motivated many researchers to work on general improvements to the core algorithm[79, 77, 149, 81, 150], and to understand its limitations[151, 152].

It is prudent to mention graph cuts[76], which solve a sub-set of the problems solved by BP. Results indicate that there is little to choose between them[80], but BP appears to be used more extensively, possibly because it is easier to work with and considerably more flexible; it also solves continuous problems, which graph cuts can not. Additionally, graph cuts are limited to solving the maximum likelihood estimate of all the random variables, when BP can optionally also solve for the posterior distribution (marginal) of each random variable. The problems of chapters 3 & 4 both require this later BP approach as a matter of tractability, and are also both continuous. Dynamic programming is also relevant for solving a very small subset of the problems that BP can solve - it is detailed by the explanation given in section C.2 however.

The following is divided into four sections - first we describe the problems that BP can be applied to, then we offer two explanations of the basic algorithm - an understandable but limited explanation followed by a more complete but harder to grasp explanation. Finally various changes to the basic algorithm are discussed.

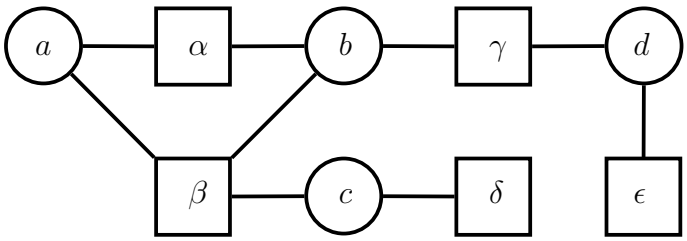


Figure C.1: An example of a factor graph. Factors are given as squares and named with Greek letters, whilst random variables are given as circles and named with Roman letters. See text for further details.

C.1 Graphical Models

Loopy belief propagation[153, p.334] is a message passing algorithm that considers an equation of the form

$$P(\mathbf{x}) = \prod_{v \in V} \psi_v(\mathbf{y}_v) \tag{C.1}$$

where $\mathbf{x}$ is a set of random variables and $\forall v; \mathbf{y}_v \subset \mathbf{x}$. Above $P(.)$ indicates a joint probability distribution over all random variables; whilst it will always have a probabilistic interpretation it is as likely to be formulated in terms of a sum of costs, which will be equivalent to the negative log of probability. There are then two standard formulations of belief propagation - sum-product, which finds the posterior distribution, or marginal, of each random variable; and min-sum (Originally called the Viterbi algorithm.), which finds the maximum likelihood assignment for all random variables. Brute forcing to calculate either of these given the above formulae is easy, but doing so for even a small problem is intractable; it is the structure in equation C.1 that allows for a fast solution. We hold off from actual implementation detail till the next section; we now consider this structure.

Equation C.1 can be represented by a bipartite graph, where each variable and each ψ function is represented as a node. The ψ nodes, called factors, are linked to the variables used in their calculation; this is called a factor graph, and an example is given in figure C.1. This example shows factors that are connected to one variable (δ, ϵ), two variables (α, γ) and three variables (β).

It is this structure that allows BP to efficiently find a solution - by sending messages along the edges of this graph. The importance of this structure is that the

more random variables a factor is connected to the longer it takes to calculate the messages - for this reason it is preferable to design a graph where the *clique* of each factor is not much more than two. In the extreme case where there is just a single factor connected to every random variable, i.e. the decomposition in equation C.1 does not exist, the algorithm becomes equivalent to brute force in terms of efficiency.

Figure C.1 contains a loop in its factor graph, which encompasses a, α, b and β. This requires that loopy belief propagation be used to solve it, for which convergence to the answer is not guaranteed[1]. When no loops exist it will obtain the correct answer; furthermore the problem can be approached without iterations or convergence detection as a suitable message update schedule will solve it in a single pass. Despite the loopy case generally not finding the best answer BP will usually converge to a good solution[127][2]. The next section gives suitable message update schedules for these scenarios, though the problems of this thesis are all loopy.

We have so far introduced the factor graph, which is but one possible graphical model; other (restricted) representations exist. The value of these alternate formulations is in part expressive, but mainly because BP can be reformulated to work directly on their graphs rather than being converted into a factor graph first. Bayesian networks use directed acyclic graphs of random variables. In such a network each node has a conditional probability distribution on it's random variable involving only the nodes the are connected to it by edges pointing at the node. No loops exist, so an exact solution may be found; this representation is, for instance, popular within expert systems. A *Markov Random Field* [MRF] is a set of random variables where a random variables probability is not (directly) dependent on every other random variable. MRFs are, by definition, expressible in the form of equation C.1, and regularly solved using BP. However, one often encounters *pairwise MRFs*, where the clique size of each factor is one or two. In

[1] In the abstract case given there is only one loop, and in consequence it will converge to the correct answer or fail to converge; it will not converge to the wrong answer, as can happen when there are multiple loops.

[2] In the case when you are solving a binary labelling problem it should be noted that graph cuts[76] will find an optimal solution for the loopy case.

this case factors connecting to one random variable, priors, are presumed to exist for all random variables, possibly as uniform distributions, and are not in the graph; factors connecting two random variables are represented as edges between the relevant random variable nodes. One major advantage of this formulation is that in reformulating BP to work on this graph there are far less nodes, which significantly reduced runtime storage requirements, and causes some speed up. The problems solved in this thesis are all expressed as pairwise MRFs, and so the reformulated version is used outside of this chapter.

C.2 Discrete Formulation

Belief propagation [BP] will initially be explained for the discrete labelling problems where we use min-sum to find the most likely assignment. This case simplifies away many details and makes understanding relatively easy. We will use negative log probabilities, which we will refer to as costs; equation C.1 therefore becomes

$$-\ln(P(\mathbf{x})) = C(\mathbf{x}) = \sum_{v \in V} \phi_v(\mathbf{y}_v) \tag{C.2}$$

where $C(.)$ is used to indicate the cost and $\phi_v(.) = -\ln \psi_v(.)$. The problem is therefore to find the assignment to the random variables that minimises $C(\mathbf{x})$:

$$\underset{\mathbf{x}}{\operatorname{argmin}}(C(\mathbf{x})) \tag{C.3}$$

We build up the explanation in steps - we initially consider the trivial single node, then we consider chains of nodes, then trees before finally considering loopy graphs.

Solving where there is only one node is trivial, but maintains the progression and serves to highlight some details assumed handled in the continuing exposition. When a single node exists all factors are limited to a clique size of one - three examples are given in figure C.2. On the left the graph has no factor - in this case we have no information as to the random variables optimal state and can consider the answer to be undefined. Alternatively we can consider this as no preference,

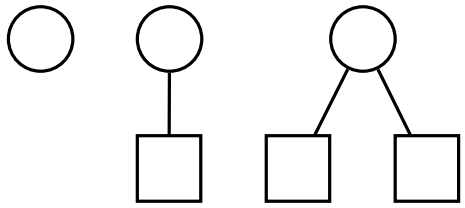


Figure C.2: Factor graphs where the solution is trivial or undefined; see text for details.

so any answer will do, or possibly all answers, in this case single states, could be iterated. In the centre of figure C.2 we show a a single factor connected to the single random variable. This factor must return a cost for every state that the random variable may be in - the optimal answer is then the state with the minimum cost. In the event of a tie the same considerations as for the no factor case apply for the tied variables; indeed, the no factor case is equivalent to the one factor case where the factor returns the same cost regardless of state. Finally we have the example on the right with two factor nodes; as indicated in equation C.2 we have to find the solution which minimises the sum of these factors. This is evidently easy as we can iterate all states and select the state that minimises this sum; from this we can see that the right case is equivalent to the centre case if we simply define the one factor as the sum of the two factors. For extended cases it should be evident that, regardless of the number, they are all equivalent to the case of a single factor on the single random variable. This idea extends further - all factors with an identical set of random variables may be merged, by simply summing the results of the separate factors for each input[3]. As a separate issue we can consider figure C.2 as a single graph with three sub-graphs - in such cases we really have three problems to be solved separately. From now on it will be assumed that factor graphs are fully connected and have no duplicate factors.

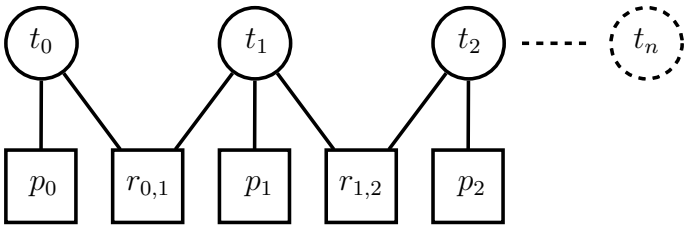


Figure C.3: A Markov chain represented using a factor graph - all possible non-duplicate factors are shown, any other factors would break the Markov chain rule. See text for further details.

C.2.1 Markov chains & Dynamic programming

A Markov chain is generally defined over time as a sequence of states where each state is dependent *only* on the last state. Time can be substituted for other variables, such as the x-axis of the page in figure C.3. In this model factors are constrained to have clique sizes of one, the priors (p), or two, the relationships between adjacent states (r). This graph is in fact a tree, however, factors with a clique of one can be ignored in this regard, and effectively merged with their associated random nodes, leaving a chain. The messages passed over this structure start at one end, t_0 and progress to the other, t_n, going via the factors $r_{n,n+1}$ at each step. Messages provide costs for each state of the involved random variable; this cost indicates the total cost of selecting that state, but only for all factors leading up to the message's point in the chain. For instance, the message passed from t_0 to $r_{0,1}$ only includes the factor p_0; the message from $r_{0,1}$ to t_1 now includes both p_0 and $r_{0,1}$, and then the message from t_1 to $r_{1,2}$ includes the state costs resulting from p_0, $r_{0,1}$ and p_1. The final message ultimately indicates the cost for the final nodes states, for all factors except for p_n, which is easily added in so the optimal state of the final node may be selected. A back-propagation method then finds the optimal state for the previous random variables.

The first message, from t_0 to $r_{0,1}$, only has to include p_0, and so can send the costs given by p_0 directly. Next message is from $r_{0,1}$ to t_1, and has to factor in the

[3]In the probabilistic case summation becomes multiplication, and is often followed by normalisation as otherwise the resulting frequency function will not sum to one. In the cost-based model normalisation does not make sense, however applying an offset to the costs has no affect, so offsetting to make the lowest cost zero is common. This is done for practical reasons, to avoid numerical overflow in large problems, and will generally be applied to the messages directly at each iteration.

costs provided by $r_{0,1}$ to its incoming message from t_0. For each state in t_1 we need to find the minimal cost possible for the previous factors. Define $r_{a,b}(s_a, s_b)$ as the cost of selecting state s_a from S_a for t_a, and s_b from S_b for t_b, as provided by the relationship factor; this minimal cost is therefore

$$M_{r \to t_b}(s_b) = \min_{S_a}(M_{t_a \to r}(s_a) + r_{a,b}(s_a, s_b)) \tag{C.4}$$

$M_{a \to b}$ is used to indicate a message, going from a to b; the subscripts of r have been omitted for clarity. For each state of the random variable receiving this message this equation calculates all possible states for the previous variable and selects the one that gives the minimum cost, where this cost factors in all previous states and factors. We now consider the message from t_1 to $r_{1,2}$, which has both an incoming message, $M_{r \to t_1}(s_1)$, as well as a prior, $p_1(s_1)$. Factoring in the prior to generate the new message is a simple case of addition however; moving to generic variables

$$M_{t_a \to r}(s_a) = M_{r \to t_a}(s_a) + p_a(s_a) \tag{C.5}$$

By applying the above message generating equations we will ultimately discover the optimal state for the final node. If we keep track at each node of which previous state resulted in the cheapest cost for each new state then we can backtrack this implied linked list to get the optimal state for each node. This procedure is in fact called dynamic programming, which is the algorithm that BP simplifies to in this case, though it has been presented in a way more typical of BP[4].

C.2.2 Trees

We now consider the case of a tree - the above algorithm adapts very simply, with the largest difference being the extension to the message passing equations needed for arbitrary clique sizes. One interpretation of the above procedure is that we are factoring out the chain so far and replacing it with a prior on the earliest node that we have not yet factored out - once the message is calculated we can continue as though all previous factors and random variables no longer exist. The message

[4] A dynamic programming representation would more typically use a trellis diagram.

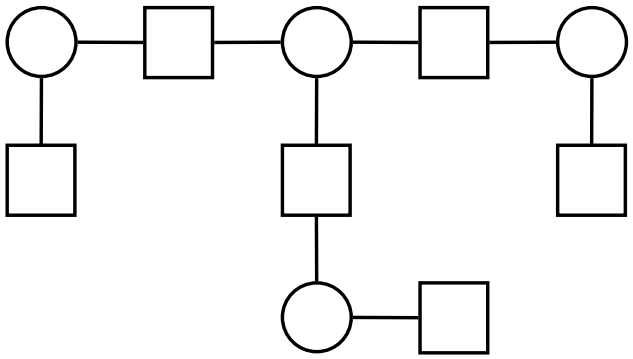


Figure C.4: A simple tree graphical model, represented as a factor graph; see text for details.

passed from a random variable to a factor is then the prior on that random variable, calculated by the summation of multiple priors as necessary. Consider the tree in figure C.4. We may remove from consideration any of the three branches to leave a simple prior on the central random variable, using the previously given algorithm. The remainder is then a chain. Alternatively we could factor out each branch separately, so instead of backtracking from a node on the edge of the graph we may backtrack from any node in the graph we choose. Taking this to its natural conclusion belief propagation on a tree is an iterative process, where for each step you select any random variable with no more than one non-prior factor (Clique size of two or more.), you then calculate the message passed to its single neighbouring random variable and pretend the random variable no longer exists, replaced by a prior. Repeated application of this process will decimate any tree to a single node, from which the best state can be selected and backtracking applied. Backtracking will now be required to track the optimal incoming state for all neighbours except the one the message is sent to.

Arbitrary clique sizes are now considered. A larger clique effectively requires the minimisation away of more variables from the joint cost function[5]. Given that a factor is connected to a set of random variables, $t \in T$, where each random variable has a set of possible states, $s_t \in S_t$ then the extended version of equation

[5]It can help to swap minimisation with marginalisation and joint cost for joint probability distribution if its more familiar, this is exactly what happens in the next subsection anyway.

C.4 that gives the message passed to variable u is

$$M_{r\to u}(s_u) = \min_{\{S_t;t\in T-u\}} \left(f(\{s_t; t \in T\}) + \sum_{t\in T-u} M_{t\to r}(s_t) \right) \tag{C.6}$$

where $\{S_t; t \in T - u\}$ indicates minimising over the states of all connected random variables except the one we are sending the message to, and likewise for the other lambda statements above. f is the factor's cost equation, and requires as input the state of all adjacent random variables. The message passed from a variable, u, to a factor is now the sum of all incoming messages, as an extension of equation C.5; we take $f \in F$ as the set of factors the variable is connected to, with g the factor we are sending the message to and get

$$M_{u\to f}(s_u) = \sum_{f\in F-g} M_{f\to u} \tag{C.7}$$

It should be noted that priors are no longer a special cases, as equation C.6 simplifies to just the cost function when the factor is connected to a single random variable. Also note that this implies we send messages to priors, which will then be ignored - this is typically optimised away. For min-sum discrete belief propagation the above equations are complete - the handling of cyclic graphs involves changes in their application only.

C.2.3 Arbitrary graphs

Initially we extend our model for trees and consider the case when we want to find the minimum cost for every possible state for every possible model. That is, we do not just want the minimal state assignment and its cost, for each state of each variable we also want the minimal cost that an assignment including that node set to that state can achieve. We refer to this as the belief of each random variable. This is valuable in assessing how strongly the graphical model considers a particular state to matter, but is also needed before we can consider the loopy case. If we have found these per variable costs then backtracking is no longer necessary, as we simply select the minimal cost state from each variable to get the

minimal cost global state. It also indicates when state selection is arbitrary, which backtracking can do but only after an increase in storage and complexity.

Consider that for the last remaining node in the tree decimating approach we have a complete cost assignment, we could therefore solve for per-state costs for the entire graph by solving the graph repeatedly, once for every variable so that each variable gets to be the last remaining. This would of course be horrifically inefficient, as many identical messages would be calculated many times between the many different runs. The reason we have this information for the last node is that it receives messages from all factors and hence has all its priors, which it may sum to obtain its final *belief*. Instead, we calculate every single possible message in a single run, so a message is passed once each way along every edge, so ultimately every variable has a complete set of priors. Initially, each node has no incoming messages, but can send a message if all but one of its messages has arrived, the message going to the node from which it has not yet received a message - this is the basis of the tree decimation approach. However, once a node receives its final message it can send a message along any of its edges, so it sends messages to all adjacent nodes except the one it has already sent a message to. By applying these extended message passing rules each random variable will ultimately have a complete set of priors incoming. It is then simple to calculate the belief at each random variable, u, as

$$B_u(s_u) = \sum_{f \in F} M_{f \to u} \tag{C.8}$$

Loopy graphs may now be considered. It is evident that when a loop occurs the above messages will not be sent as the conditions for sending will never be satisfied. This is resolved by sending messages all the time, iteratively[6]. Messages that have not yet been received are simply set as constant[7]. Given a tree as input and then iterating sending all messages it will obviously converge to the correct solution eventually, however, it can also produce reasonable answers for loopy

[6]This can be done in loopy regions only, with trees handled as previously.

[7]Usually as 0's, so they do not increase the cost unnecessarily. Costs should also not be negative, as this can result in runaway loops that always reduce the cost, causing representational issues. Offsetting messages so that the lowest cost is zero is a common means of preventing the cost heading skyward, and removing negative costs if they exist. As a result final costs are relative rather than absolute, but for many problems this is satisfactory.

graphical models. Loopy graphs will not always converge however, and it is necessary to decide what to do in such cases - either indicating that the problem is taking too long or taking the solution at the final iteration anyway. It is possible for sub-parts to have converged whilst other parts have not, so may provide useful information. More critically, if there is more than one loop it will not necessarily converge to the right answer. Practical results indicate however that it will usually get a good answer, and theoretically results show that it will be a maxima within a certain region of the state space, which can be very large[151].

C.3 Continuous Formulation

The above description is mostly complete, with very little additional material required to handle the two remaining cases of continuous distributions and sum-product belief propagation. We start by giving the continuous sum-product equations, first by replacing the equation for passing a message from a factor to a variable, equation C.6, with

$$M_{r\to u}(s_u) \propto \int_{\{S_t;t\in T-u\}} \left(f(\{s_t; t \in T\}) \prod_{t\in T-u} M_{t\to r}(s_t) \right) \tag{C.9}$$

Now the replacement for equation C.7, variable to factor

$$M_{u\to f}(s_u) \propto \prod_{f\in F-g} M_{f\to u} \tag{C.10}$$

And, finally, the replacement for equation C.8, which extracts the final belief after convergence

$$B_u(s_u) \propto \prod_{f\in F} M_{f\to u} \tag{C.11}$$

A simple comparison will reveal that we have replaced addition with multiplication and minimisation with summation, where we use the integral as it is continuous. The detail omitted is that we are now working with probabilities rather than costs, so the messages sent and the result of the belief equation are all probability distributions; because of this the above equations have omitted

normalisation terms, however, in practise normalisation can be ignored until calculating the final output (Though normalisation may be performed during run-time, for numerical stability.). Consider why we have made the changes we have. Swapping multiplication for addition is entirely a result of taking the logarithm of probability. Indeed, we can define the max-product algorithm, which is identical to the min-sum algorithm except it works with probability distributions rather than costs (The negative logarithms of probability.) - moving to costs is mostly for computational and explanatory convenience. This highlights the real difference between the sum-product and the min-sum algorithm - the sum-product algorithm sums distributions, and hence calculates the marginal, whilst the min-sum algorithm is interested in only the most likely option - this is why the algorithms give the answers that they do. Switching from summation to the integral is simply a consequence of moving from a discrete to continuous formulation. With these observations all basic forms of BP should now be understood by the reader - the next subsection considers various possible changes for completeness, though most of them are not used nor necessary in the later chapters of this thesis.

C.4 Changes & Improvements

Changes to basic BP can be categorised into three groups - structural, message calculation and message representation. This last group is not really a change, as the basic formulation does not specify the means of representation, but certain representations come with specific considerations and advantages. We will now iterate some examples of these, in the reverse of the above order.

C.4.1 Message Representation

In the discrete case messages are typically represented by arrays of real numbers, where each entry corresponds to the probability or cost of a possible state[8]. The interesting cases are for continuous problems however. A textbook parametric

[8]Other encodings are used in certain cases however. For instance, if there are a large number of possible states and most of them will have the same value, due to say a cost cap (Useful for robustness.), a certain amount of simple compression can be achieved by storing only the values that deviate.

distribution is probably the simplest choice. We use the Gaussian for this purpose in chapter 3. In the Gaussian case multiplication and integration are both easy, as long as the factors can be represented as Gaussians, however, for most distributions this is not the case, or they are not capable of reasonably approximating the data in question. This results in two alternative models being popular, specifically mixtures and piecewise graphs. A mixture is a probability distribution represented as a sum of simpler weighted distributions, for instance a Gaussian mixture consists of a number of Gaussian distributions, where the PDF at each point is defined by summing these distributions PDF's multiplied by their weight. Theoretically any arbitrary distribution can be represented, and multiplication is well defined if the base distribution has a multiplication operation. The problem is that each multiplication will produce a new distribution with an exponentially increasing number of terms, so terms have to be pruned using some mechanism; often a Monte-Carlo approach (particle filtering) is taken to estimate a new distribution entirely, as storing and sampling the multiplied out distribution would be too resource intensive. In such an approach integration is not a problem; a good example of this technique can be found in Sudderth et al.[149]. A piecewise graph can be used to represent a probability distribution quite simply - usually as a piecewise constant graph (A variable width histogram) or a piecewise linear graph, though other systems are plausible. See Potetz[26] for an example with variable width histograms. Multiplication in such cases is trivial, as in fact are all the operations that BP can require - this method has the advantage of working with min-sum and costs also. Again, extra terms will generally be added with each operation, however, this increase is linear and the representation far more amenable to being pruned in a sensible way.

C.4.2 Message Calculation

Consider passing a message in the discrete case. Examining the factor to variable message passing equation, C.9, it is evident that for each state in the output message we have to find the sum or minimum of $\prod_{t \in T-u} \#S_t$ terms, and that we have to do this $\#u$ times. Put bluntly, if all variables have n states then it is

an $O(n^r)$ operation to calculate each message, where r is the number of random variables the factor is connected to. This has to be performed r times, and has to be done for every factor in the graph. If f is the number of factors, and they are all connected to r random variables, of which each has n states the total computation per iteration is therefore $O(frn^r)$. This quickly becomes unmanageable for large problems; fortunately there is structure to be exploited, which allows certain problems to be solved much quicker.

A simple example of this is given by Felzenszwalb & Huttenlocher[79], for the case of a factor connected to two variables where the cost is a convex function on the difference between the variables. This implies a relationship between adjacent states in the outgoing message that 'blocks' states that are not adjacent from mattering in the calculation - as a consequence only adjacent states need to be considered in a two pass algorithm, which changes a $O(n^2)$ algorithm into a $O(n)$ algorithm. They demonstrate this case for a stereopsis algorithm, where the states represent discrete disparities and the factors between them are an increasing cost for a change in disparity, so the algorithm prefers fronto-parallel surfaces.

Linear constraint nodes, as introduced by Potetz[26], are a more sophisticated example, designed to simplify cases where a factor is connected to many variables. It requires that factors take on a specific form

$$\psi_v(\mathbf{Y}_v) = \prod_{i \in I} g(\mathbf{y}_v \cdot \mathbf{s_i}) \tag{C.12}$$

where $\mathbf{y}_v$ is the vector of states and $\mathbf{s_i}$ an arbitrary vector. This allows a separation of variables to take place, which means that each sum/integration/max can be performed separately. Consequentially, the message passing calculation is converted from being $O(n^r)$ to $O(Irn^2)$, which is much quicker to compute. By using a product of linear constraint nodes most arbitrary factor functions can be approximated, making this a very powerful technique.

C.4.3 Structural

The biggest improvements to BP come from fundamentally changing or enhancing the algorithm. These improvements mostly come from a thorough understanding of how the algorithm works, leading to a useful generalisation. Loopy BP actually converges to extrema of the Bethe free energy[154, 127], a concept from physics, which is an approximation of the actual free energy. Bethe free energy is the simplest version of the Kikuchi approximations. It is this observation that allowed Yedidia, Freeman & Weiss[154, 127] to generalise the algorithm to allow a trade off between speed/memory consumption against the accuracy of the result, by implementing versions that optimise for any of the Kikuchi free energy approximations. We can consider basic BP to involve passing messages between regions, where each region consists of a random variable and all random variables it is connected to via a factor node. Regions then exchange messages if they share a random variable, the messages giving the belief on the shared random variable and taking care to not include information received from the region the message is being sent to. *Generalized belief propagation* allows for arbitrary connected regions, including regions that are sub-sets of others and region pairs where the overlap includes multiple random variables. Complexity is increased, with effort required to avoid over-counting overlapping regions between the sets - as a consequence the algorithm consumes more memory and takes longer. The advantage is that the algorithm can converge to an answer much closer to the global minima. One way of looking at this is as introducing a graduation of algorithms between the original BP formulation and brute force.

Convergence of BP does not always occur, despite a proof that there is always a lower bound on the energy being minimised and therefore a fixed point to find[154]. Several authors have proposed modifications to ensure convergence[150, 155]. A particularly simple, and relatively fast, example is Heskes, Albers & Kappen[155]. They introduce a change to the message calculation equations that ensures convergence in all cases. It is a generalisation of the work of Yuille[150] that works by decomposing the cost function into convex and concave components. The flaw with the approach is its only valid for the sum-product algorithm.

If you want the min-sum output but can only calculate the sum-product result, as above, then you can introduce a temperature term[127]. This simply involves taking the factor functions to the power of $1/T$ - T starts high and is reduced as the algorithm continues to run, forcing the algorithm to approximate the maximum likelihood estimate.

A final consideration is the message update schedule - traditionally in loopy BP a message is sent by every node every iteration. This is not always necessary, for instance if the messages being sent to a node are not changing then the messages the node sends out will not be changing, and calculating them is a waste of time. There is also a dependency consideration - each message sent is dependent on messages from the previous time step, which means that in principal we need to store two time-steps worth of messages. This is not always required, for instance a grid can be updated using already sent messages in sweeps in each of the four directions - this will accelerate convergence. Another approach for a grid, when random variables are only connected to their four-way neighbours is given by Felzenszwalb & Huttenlocher[79]. They only update half the variables in each iteration, in a checker-board pattern - this means that the incoming messages of each variable have not been updated when it is calculating its message and no dependency issue exists. This approach and similar once are simple but useful - in this case it halves memory consumption and doubles the speed of convergence.

Appendix D
Directional Statistics

CHAPTERS 4 and 5 utilise directional statistics, where distributions on surface orientation are needed. Directional statistics[135] is a mostly overlooked field - this is unfortunate in computer vision research as there are many instances of directional data. The few works that do exist mostly use the von-Mises-Fisher[135, p.36] distribution. For instance Hara et al.[116] substitute it for the Gaussian in the Torrence-Sparrow illumination model, this modified model then proves tractable when solving for a mixture model, for the purpose of inferring scene illumination. Other examples include Calderara et al.[156] using it to detect anomalous behaviours in humans and Banerjee et al.[157] using it for clustering.

In chapter 4 we use the Fisher-Bingham distribution[158], of which we can find no mention in the computer vision literature; we also use it within a belief propagation framework. Using a directional distribution, the von-Mises-Fisher, with belief propagation was explored by Sudderth[147] for hand tracking, in a followup to his work on non-parametric belief propagation[149]. Rotations in the model are represented by quaternions, and a mixture of 4D von-Mises-Fisher distributions is used, with duplicated entries to handle the negation of a quaternion being a no-op. This is an expressive distribution on rotation, used to model the rotations between joints in the model. Integration into the belief propagation framework is done by sampling within a particle filtering approach, extended to arbitrary graphs. In chapter 4 we take a more analytical approach, which is necessary as the sampling approach used by Sudderth, whilst suitable for small models, is unsuitable for large models with millions of nodes due to resource usage.

In the following sections we consider basic concepts, the directional distributions relevant to this work, and various operations we will need to perform on them.

D.1 Concepts

The application of typical statistical methods to angular quantities fails. To give an example take the angles 1° and 359° - if we ignore that they are angles and take the mean we end up with a value of 180°, which is as far away from both as you can get on a circle. 0° is the right answer, and we can get it by expressing 359° as -1° or 1° as 361°, but such fiddling only works in the general case if we choose a split point directly opposite the mean, which implies we already know it.

Knowing that traditional statistics will fail we need an alternate system, but first some consideration of the problems this will allow us to solve is in order. An angle is a number modulus 360, and any other quantity that wraps around warrants the application of directional statistics. This can include time of day or days of week, for instance. In computer vision the angles between components of deformable models and the orientations of an edge or a texture are plausible examples. So far we have considered 1D angles, but it is easy to extend to higher dimensions. This includes the 2 degrees of freedom of directions in $3D$ space, which is where this work comes in by probabilistically modelling surface orientation. In $4D$ space a direction can represent a rotation via a quaternion, which has the previously mentioned use in deformable models[147]. As an example of d dimensional space we can consider a simplex (Generalised triangular coordinates), re-normalised to be of length 1 rather than to sum to 1. To conclude there are many areas where directional statistics are relevant.

Now we return to the mean. An angle can be represented by a $2D$ unit length vector, i.e.

$$\theta \leftrightarrow \begin{pmatrix} \cos(\theta) \\ \sin(\theta) \end{pmatrix} \tag{D.1}$$

If we take such vectors, sum them and then re-normalise before finally converting back to an angle we get the correct mean, i.e. given a set of angles Θ then their

mean is[135, p.15]

$$\bar{\theta} = \tan^{-1}\left(\frac{\sum_{\theta\in\Theta}\sin(\theta)}{\sum_{\theta\in\Theta}\cos(\theta)}\right) \tag{D.2}$$

assuming $\tan$ handles the quadrants correctly. This extends to 3 dimensions using spherical coordinates, and onwards to arbitrary dimension. Standard deviation is another common summary statistic, however, whilst an intuitive conversion for the mean exists this is not the case for the standard deviation. Instead we introduce the *mean resultant length*, which is the Euclidean length of the sum of the angles in vector form divided by the number of samples[135, p.15]

$$R = \frac{\sqrt{\left(\sum_{\theta\in\Theta}\sin(\theta)\right)^2 + \left(\sum_{\theta\in\Theta}\cos(\theta)\right)^2}}{\#\Theta} \tag{D.3}$$

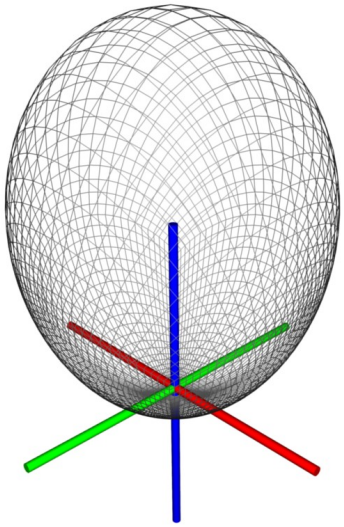
Resultant length has the range $R \in [0, 1]$, where 0 equates to the uniform distribution and 1 equates to every sample having the same direction.

D.2 Distributions

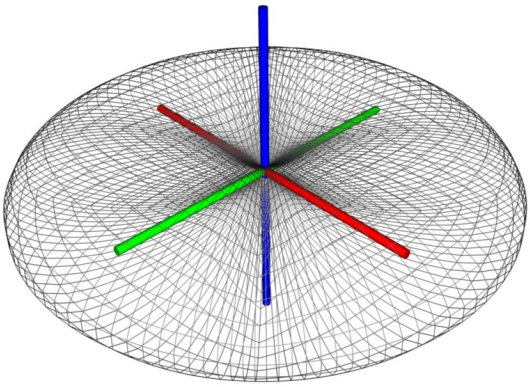
There are three main classes of parametric, continuous distributions used in directional statistics - *wrapped*, *projected* and *constrained*[1]. A wrapped distribution is any distribution from traditional statistics defined to wrap around, so that any sample of the probability density function [PDF] is the sum of the wrapped distribution every $360°$, for infinity. A projected distribution uses the vector representation of direction, defining the direction of each vector $\mathbf{x}$ as $\mathbf{x}/|\mathbf{x}|$. The PDF of a distribution on arbitrary vectors from classical statistics is then integrated over for all vectors of a given direction, to get the probability of that given direction. We will now ignore these, as our interest is in the, much more convenient, constrained distributions.

A constrained distribution works with the vector representation of a direction, taking a classical statistics distribution on the vectors and constraining it to unit length vectors only. Unlike a projected distribution where you have to integrate over all points in the original sample space that project to a given direction here

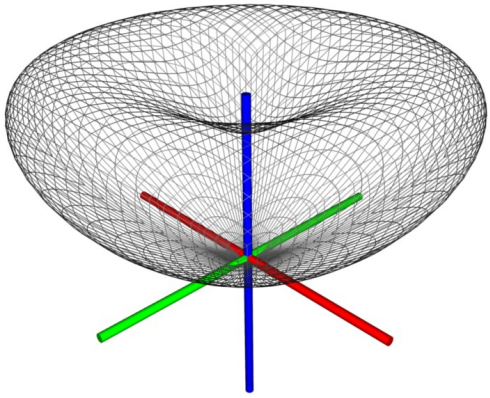
[1] As an example of a discrete distribution consider the lattice distribution, which is n locations equally spaced around the circle, weighted. Most of the methods for dealing with this transfer directly from normal statistics.



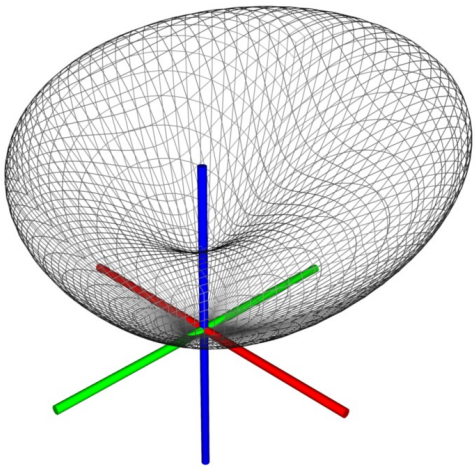
(a) Fisher distribution



(b) Bingham distribution



(c) Bingham-Mardia distribution



(d) Fisher-Bingham distribution

Figure D.1: Representative directional distributions, represented by surfaces where greater distance from the origin (The primary-colour axis) equals greater probability for that direction. i.e. for the Fisher distribution, D.1(a), you can see a maxima at the top where its far from the origin in the up blue axis direction and a minima at the bottom where its close to the origin, in the down blue axis direction. Mathematically, the distance of each point on the sphere is proportional to the PDF of the distribution for the direction from the origin to that point.

you just sample a single point. As an example, the von-Mises distribution[135, p.36] is a $2D$ Gaussian distribution with an arbitrary mean and the co-variance matrix a multiple of the identity matrix. By restricting the Gaussian to this form where it only applies to normalised vectors it may then be re-parametrised as the PDF

$$P_{vM}(\hat{\mathbf{x}}; \mathbf{u}) \propto \exp(\mathbf{u}^T \hat{\mathbf{x}}) \tag{D.4}$$

where $\hat{\mathbf{x}} \in \Re^2, |\hat{\mathbf{x}}| = 1$ is the considered direction and $\mathbf{u} \in \Re^2$ is the parameter. If $\mathbf{u}$ is $[0, 0]^T$ then you have the uniform distribution, otherwise $\mathbf{u}/|\mathbf{u}|$ indicates the direction with highest probability, whilst $-\mathbf{u}/|\mathbf{u}|$ indicates the direction with the lowest probability. We define concentration as $k = |\mathbf{u}|$, the length of $\mathbf{u}$ - the higher the concentration the more the distribution is concentrated around $\mathbf{u}/|\mathbf{u}|$. Indeed, given this it can be observed that the distribution can also be expressed as

$$P_{vM}(\hat{\mathbf{x}}; \mathbf{u}) \propto e^{k \cos\theta} \tag{D.5}$$

where θ is the angle between the two directions. Using this observation Fisher[159] extended the idea to projecting a $3D$ distribution for directions in $3D$ space - the definition of the Fisher distribution, $P_F(.)$ is in fact identical except it uses $3D$ vectors rather than $2D$ vectors[2]. Figure D.1(a) is an exemplar Fisher distribution. It should be obvious that this can be extended to arbitrary dimensionality, and in the n dimensional case it is referred to jointly, as the von-Mises-Fisher[135, p.159] distribution.

Normalisation of the PDF has so far been ignored. Given the d dimension von-Mises-Fisher distribution then[135, p.168]

$$P^d_{vMF}(\hat{\mathbf{x}}; \mathbf{u}) = \frac{(|\mathbf{u}|/2)^{d/2-1}}{\Gamma(d/2) I_{d/2-1}(|\mathbf{u}|)} \exp(\mathbf{u}^T \hat{\mathbf{x}}), \qquad \mathbf{u}, \hat{\mathbf{x}} \in \Re^d, |\hat{\mathbf{x}}| = 1 \tag{D.6}$$

where $\Gamma(.)$ is the gamma distribution and I_x is the modified Bessel function of the first kind, order x. Rather conveniently, in the Fisher case, when $d = 3$, it simplifies

[2]The original equations did not use vectors and instead used angles directly, making this extension seem far less obvious. The vector based formulation was adopted by Fisher.

and we get

$$P_F(\hat{\mathbf{x}};\mathbf{u}) = \frac{|\mathbf{u}|}{\texttt{sinh}(|\mathbf{u}|)}\texttt{exp}(\mathbf{u}^T\hat{\mathbf{x}}) \tag{D.7}$$

In much the same way that as a sample from a Gaussian approaches infinite size the Gaussian defined using the mean and standard deviation of that sample should approach the distribution the sample is drawn from we can calculate a von-Mises-Fisher distributions from summary statistics. Specifically[135, p.197], we set the mean direction, equation D.2, as the direction of $\mathbf{u}$ and calculate the concentration as $k = A_d^{-1}(R)$, where R is the resultant length from equation D.3, n the number of samples and $A_d(k)$ is

$$A_d(k) = \frac{I_{d/2}(k)}{I_{d/2-1}(k)} \tag{D.8}$$

We will now consider several further distributions, all used in later chapters, all based on the Gaussian, with their normalisation omitted until section D.3.

The *Bingham distribution*, which is exemplified in figure D.1(b), is based on a Gaussian where the mean is set to the zero vector but the co-variance matrix is unconstrained,

$$P_B(\hat{\mathbf{x}};\mathbf{A}) \propto \exp(\hat{\mathbf{x}}^T\mathbf{A}\hat{\mathbf{x}}) \tag{D.9}$$

where $\mathbf{A}$ is a $d \times d$ symmetric matrix, $\mathbf{A} = \mathbf{A}^T$; it is symmetric because it is the inverse of a covariance matrix. We now consider the eight parameter Fisher-Bingham distribution[158][3] [FB_8]. This distribution is, quite simply, the multiplication of the von-Mises-Fisher distribution and the Bingham distribution, and is given by

$$P_{\mathrm{FB}_8}(\hat{\mathbf{x}};\mathbf{u},\mathbf{A}) \propto \exp(\mathbf{u}^T\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\mathbf{A}\hat{\mathbf{x}}) \tag{D.10}$$

It is a conditioned Gaussian, with no constraint on its parameters; it also forms the basis for the work of chapter 4, so we will now go into some detail, in the following paragraphs and subsections. An example is given in figure D.1(d).

[3]There is also the five parameter version, usually called the Kent distribution, which restricts the parameters[135, p.176] such that the mean direction is a multiple of the smallest eigenvector of the $\mathbf{A}$ matrix. These parameter counts only apply if we are assuming a $3D$ FB_8 distribution - the distribution is in fact general and applies to all dimensions of 3 and greater. In the 2 dimensional case it works, but is equivalent to the double-angle Bingham distribution.

For convenience we may represent the FB_8 distribution as

$$\exp(\mathbf{u}^T\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\mathbf{A}\hat{\mathbf{x}}) = \Omega[\mathbf{u}, \mathbf{A}] \tag{D.11}$$

Using this notation multiplication is

$$\Omega[\mathbf{u}, \mathbf{A}]\Omega[\mathbf{v}, \mathbf{B}] = \Omega[\mathbf{u} + \mathbf{v}, \mathbf{A} + \mathbf{B}] \tag{D.12}$$

This is one of the key advantages of using a conditioned distribution - multiplication remains identical to the unconditioned distribution i.e. if the conditioned distribution has a multiplication operation, as the Gaussian has, then so does the conditioned distribution. Consequentially, this is simply the multiplication of two Gaussians, but with a different representation.

We may decompose the FB_8 distribution. As $\mathbf{A}$ is symmetric we may apply the eigen-decomposition to obtain $\mathbf{A} = \mathbf{B}\mathbf{D}\mathbf{B}^T$, where $\mathbf{B}$ is orthogonal and $\mathbf{D}$ diagonal. This allows us to write

$$P_{\mathrm{FB}_8}(\hat{\mathbf{x}}; \mathbf{u}, \mathbf{A}) \propto \exp(\mathbf{v}^T\hat{\mathbf{y}} + \hat{\mathbf{y}}^T\mathbf{D}\hat{\mathbf{y}}) \tag{D.13}$$

where $\mathbf{v} = \mathbf{B}^T\mathbf{u}$ and $\hat{\mathbf{y}} = \mathbf{B}^T\hat{\mathbf{x}}$. As $|\hat{\mathbf{y}}| = 1$ we may offset $\mathbf{D}$ by an arbitrary multiple of the identity matrix, this allows any given entry to be set to 0. For the $3D$ scenario, $\hat{\mathbf{x}}, \mathbf{u} \in \Re^3$, we can therefore consider it the case that $\mathbf{D} = \mathrm{Diag}(\alpha, \beta, 0)$, with $\alpha > 0$ and $\beta > 0$ (Possibly requiring the use of a permutation matrix.) such that

$$P_{\mathrm{FB}_8}(\hat{\mathbf{x}}; \mathbf{u}, \mathbf{A}) \propto \exp(\mathbf{v}^T\hat{\mathbf{y}} + \alpha\hat{\mathbf{y}}_x^2 + \beta\hat{\mathbf{y}}_y^2) \tag{D.14}$$

The decomposition is used in later sections, when finding the maxima for instance (Section D.4).

We now introduce one final distribution, which is a sub-distribution of the FB_8 distribution, i.e. it is a constraint on the form of the FB_8 distribution. It is the *Bingham-Mardia distribution*[160], which is exemplified in figure D.1(c), and using

the introduced notation for the FB_8 distribution it is given by

$$\exp(-k(\hat{\mathbf{u}}^T\hat{\mathbf{x}} - \cos\theta)^2) = \Omega[2k\cos(\theta)\hat{\mathbf{u}}, -k\hat{\mathbf{u}}\hat{\mathbf{u}}^T] \tag{D.15}$$

where $\hat{\mathbf{u}}$ is the direction of the axis of a cone and θ the angle of that cone. This distribution has a small circle as its maximum, i.e. a circular maximum at an arbitrary point on the sphere, for the $3D$ case. This makes it particularly useful for SfS, as will be shown in chapter 4.

D.3 FB_8 normalising constant

The normalisation of the FB_8 distribution is a tricky problem, best avoided if at all possible. Like most distributions, it involves solving the integral of the PDF for all directions, specifically the equation is, in the d dimensional case

$$P_{\mathrm{FB}_8}(\hat{\mathbf{x}}; \mathbf{u}, \mathbf{A}) = \frac{\exp(\mathbf{u}^T\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\mathbf{A}\hat{\mathbf{x}})}{\int_{\hat{\mathbf{x}}\in\Re^d, |\hat{\mathbf{x}}|=1} \exp(\mathbf{u}^T\hat{\mathbf{x}} + \hat{\mathbf{x}}^T\mathbf{A}\hat{\mathbf{x}})} \tag{D.16}$$

There is no known analytical solution to the integral and so it has to be approximated.

For completeness we now give the approximation of Kume & Wood[161], which uses a saddlepoint approximation[4]. Several versions are given, all of the reciprocal of the integral, of which the first is[161, Eq. 15]

$$\sqrt{2}\pi^{(d-1)/2}(\hat{\mathbf{K}}_\theta^{(2)} \left(\prod_{i=1}^{d} \frac{1}{\sqrt{\lambda_i - \hat{t}}}\right) \exp\left(-\hat{t} + \frac{1}{4}\sum_{i=1}^{d} \frac{\gamma_i^2}{\lambda_i - \hat{t}}\right) \tag{D.17}$$

where d is the dimensionality, λ the eigenvalues of A and γ the components of $\hat{\mathbf{y}}$, from equation D.13. $\hat{t}$ is the solution to $\hat{\mathbf{K}}_\theta^{(1)}(\hat{t}) = 1$, where[161, Eq. 9]

$$\hat{\mathbf{K}}_\theta^{(1)}(t) = \sum_{i=1}^{d} \left\{ \frac{1}{2(\lambda_i - t)} + \frac{\gamma_i^2}{4(\lambda_i - t)^2} \right\} \tag{D.18}$$

which is the first order saddlepoint density approximation. The second order is

[4]They also give in the same paper an approximation for the Bingham distribution, which is similarly inconvenient to work with.

also used, which is the $j = 2$ case of[161, Eq. 10]

$$\hat{\mathbf{K}}_\theta^{(j)}(t) = \sum_{i=1}^{d} \left\{ \frac{(j-1)!}{2(\lambda - t)^j} + \frac{j!\gamma_i^2}{4(\lambda_i - t)^{j+1}} \right\} \tag{D.19}$$

Further improvements are given, of which the best performing is to multiply the original estimate by $\exp(\hat{\rho}_4/8 - 5\hat{\rho}_3^2/24)$, where

$$\hat{\rho}_j = \frac{\hat{\mathbf{K}}_\theta^{(j)}(\hat{t})}{(\hat{\mathbf{K}}_\theta^{(2)}(\hat{t}))^{j/2}} \tag{D.20}$$

This improved version is used when a normalisation constant is needed in the work of chapter 4.

D.4 FB_8 maxima

In chapter 4 finding the maxima of the FB_8 distribution is important - the presented algorithm needs them for the $3D$ case of surface orientation. A literature search turned up no specific means of solving this problem, though an iterative algorithm would obviously work. The problem is that we have to find the maxima for a different distribution for each pixel in an image, and an iterative approach needs multiple starts as the FB_8 distribution usually has two maxima, but can also have a small circle of maxima, which causes problems. It simply takes too long, and in fact dwarfs the computation time of the core algorithm, to take a naive iterative approach. For this reason we now propose a slightly more intelligent and much more efficient approach to this problem, built on converting this problem into the distance to an ellipsoid from a point problem, which has the same critical points.

We start from the decomposed $3D$ FB_8 distribution, (D.13). The FB_8 distribution is a conditioned multivariate normal distribution[135, p. 175]

$$\Sigma = \frac{-(\mathbf{D} + c\mathbf{I}_3)^{-1}}{2} \qquad \bar{\mu} = \Sigma\mathbf{v} \tag{D.21}$$

where $c\mathbf{I}_3$ is a scaled identity matrix selected to make $\mathbf{D} + c\mathbf{I}_3$ negative definite; this may be done as the condition on $\mathbf{y}$ means this is equivalent to multiplying

the distribution prior to the normalisation term. The critical points are therefore the critical points on the unit sphere of the distance function to $\bar{\mu}$ when using Mahalanobis distance, which is

$$\sqrt{(\hat{\mathbf{x}} - \bar{\mu})^T \Sigma^{-1} (\hat{\mathbf{y}} - \bar{\mu})} \tag{D.22}$$

Since $\mathbf{D}$ is diagonal Σ is diagonal, so we may rewrite equation D.22 as

$$\sqrt{\sum_i [(\hat{\mathbf{y}}_i - \bar{\mu}_i)^2 \sigma_i^{-1}]} \tag{D.23}$$

where $\sigma_i = \Sigma_{ii}, i \in \{1, 2, 3\}$. This is rearranged as

$$\sqrt{\sum_i [(\mathbf{z}_i - \sqrt{\sigma_i^{-1}} \bar{\mu}_i)^2]} \tag{D.24}$$

where $\mathbf{z}_i = \sqrt{\sigma_i^{-1}} \hat{\mathbf{y}}_i$. This is now Euclidean distance when solving for $\mathbf{z}_i$, and the constraint that $\hat{\mathbf{y}}$ be of unit length becomes the equation of an ellipsoid

$$\sum_i [(\frac{\mathbf{z}_i}{\sqrt{\sigma_i^{-1}}})^2] = 1 \tag{D.25}$$

We use the method of Hart[162] to solve the closest point on an ellipsoid to a point problem. This expresses the distance as an order six polynomial, such that the roots are points on the ellipsoid where the line passing through them and $\bar{\mu}$ is coincident with surface orientation. The roots then correspond to the FB_8 critical points. It is a simple matter to select the two maxima. Instead of always finding all roots we make use of the initialisation given by Hart[162] when finding the minimum distance. This derives from the observation that when $\bar{\mu}$ is outside the ellipsoid, and therefore outside the sphere before the transformation, there can only be two real roots which are the smallest and largest roots, and hence the closest point and furthest point. These may be found using Newton iterations initialised at distances where they are past the extrema, which can be selected using the minimum and maximum distance to a sphere that contains the ellipsoid, i.e. the sphere of radius $\max_i(\sqrt{\sigma_i^{-1}})$. In the case that $\bar{\mu}$ is within the sphere we

revert to an eigenvalue method and find all roots, from which we extract the maxima. This technique is of course iterative, just like the problematic approach mentioned earlier, however, by converting to the well known polynomial root finding problem it can be done much quicker, and as the roots are in a known relationship converging to them does not require multiple restarts. The case that the distribution has an infinite ring of maxima is also easy to detect.

Bibliography

[1] G. F. Poggio, F. Gonzalez, and F Krause. Stereoscopic mechanisms in monkey visual cortex: Binocular correlation and disparity selectivity. *Journal of Neuroscience*, 8(12):4531–4550, 1988.

[2] V. S. Ramachandran. Perception of shape from shading. *Nature*, 331:163–165, 1988.

[3] Y. Yang and A. Yuille. Sources from shading. *Computer Vision and Pattern Recognition*, pages 534–539, 1991.

[4] Y. G Leclerc and A. F. Bobick. The direct computation of height from shading. *Computer Vision and Pattern Recognition*, pages 552–558, 1991.

[5] J. E. Cryer, P. S. Tsai, and M. Shah. Integration of shape from shading and stereo. *Pattern recognition*, 28(7):1033–1043, 1995.

[6] B. K. P. Horn. *Shape From Shading: A Method For Obtaining The Shape Of A Smooth Opaque Object From One View*. PhD thesis, Massachusetts Institute of Technology, 1970.

[7] J. Van Diggelen. A photometric investigation of the slopes of the heights of the ranges of hills in the maria of the moon. *Bulletin of the astronomical institutes of the Netherlands*, pages 283–289, 1951.

[8] T. Rindfleisch. Photometric method for lunar topography. *Photogrammetric Engineering*, pages 262–276, 1966. Original could not be obtained, however a report version was submitted to NASA's JPL, ref. 32-786.

[9] R. Zhang, P-S. Tsai, J. E. Cryer, and M. Shah. Shape from shading: A survey. *Pattern Analysis and Machine Intelligence*, 21(8):690–706, 1999.

[10] K. M. Lee and C-C. J. Kuo. Shape from shading with perspective projection. *CVGIP: Image Understanding*, 59(2):202–212, 1994.

[11] Q. Zheng and R. Chellappa. Estimation of illuminant direction, albedo, and shape from shading. *Pattern Analysis and Machine Intelligence*, 13(7):680–702, 1991.

[12] M. Bichsel and A. P. Pentland. A simple algorithm for shape from shading. *Computer Vision and Pattern Recognition*, pages 459–465, 1992.

[13] C-H. Lee and A. Rosenfeld. Improved methods of estimating shape from shading using the light source coordinate system. *Artificial Intelligence*, 26(2):125–143, 1985.

[14] A. Pentland. Shape information from shading: A theory about human perception. *Computer Vision*, pages 404–413, 1988.

[15] P-S. Tsai and M. Shah. Shape from shading using linear approximation. *Image and Vision Computing*, 12:487–498, 1994.

[16] J-D. Dourou, M. Falcone, and M. Sagona. Numerical methods for shape from shading: A new survey with benchmarks. *Computer Vision and Image Understanding*, 109(1):22–43, 2008.

[17] K. Ikeuchi and B. K. P. Horn. Numerical shape from shading and occluding boundaries. *Artificial Intelligence*, pages 141–184, 1981.

[18] M. J. Brooks and B. K. P. Horn. Shape and source from shading. *Artificial Intelligence*, pages 932–936, 1985.

[19] R. T. Frankot and R. Chellappa. A method for enforcing integrability in shape from shading algorithms. *Pattern Analysis and Machine Intelligence*, 10(4):439–451, 1988.

[20] E. R. Hancock A. Robles-Kelly. A graph-spectral approach to shape-from-shading. *Image Processing*, 13(7):912–926, 2004.

[21] P. L. Worthington and E. R. Hancock. New constraints on data-closeness and needle map consistency for shape-from-shading. *Pattern Analysis and Machine Intelligence*, 21(12):1250–1267, 1999.

[22] M. Louw and F. Nicolls. A loopy belief propagation approach to the shape from shading problem. *International Conference on Computer Vision Theory and Applications*, 2007.

[23] M. Louw and F. Nicolls. A spatially multiresolution, MRF optimization based approach to the shape from shading problem. *Visualization, Imaging, and Image Processing*, 2007.

[24] P. L. Worthington. Reillumination-driven shape from shading. *Computer Vision and Image Understanding*, 98(2):325–343, 2005.

[25] P. L. Worthington. Re-illuminating single images using albedo estimation. *Pattern Recognition*, 38(8):1261–1274, 2005.

[26] B. Potetz. Efficient belief propagation for vision using linear constraint nodes. *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[27] Potetz, private correspondence, 2008-08-07.

[28] J. J. Atick, P. A. Griffin, and A. N. Redlich. Statistical approach to shape from shading: Reconstruction of three-dimensional face surfaces from single two-dimensional images. *Neural Computing*, 8(6):1321–1340, 1996.

[29] R. Dovgard and R. Basri. Statistical symmetric shape from shading for 3d structure recovery of faces. *European Conference on Computer Vision*, 2:99–113, 2004.

[30] W. A. P. Smith. *Statistical Methods For Facial Shape-from-shading and Recognition*. PhD thesis, University of York, 2007.

[31] G. Healey and T. O. Binford. Local shape from specularity. *Computer Vision, Graphics, and Image Processing*, 42(1):62–86, 1988.

[32] S. Bakshi and Y-H. Yang. Shape from shading for non-lambertian surfaces. *Image Processing*, 2:130–134, 1994.

[33] O. Vega and Y. H. Yang. Shading logic: A heuristic approach to recover shape from shading. *Pattern Analysis and Machine Intelligence*, 15(6):592–597, 1993.

[34] A. H. Ahmed and A. A. Farag. A new formulation for shape from shading for non-lambertian surfaces. *Computer Vision and Pattern Recognition*, 2:1817–1824, 2006.

[35] M. Oren and S. K. Nayar. Diffuse reflectance from rough surfaces. *Computer Vision and Pattern Recognition*, pages 763–764, 1993.

[36] H. Ragheb and E. R. Hancock. Separating lambertian and specular reflectance components using iterated conditional modes. *British Machine Vision Conference*, 2001.

[37] K. M. Lee and C-C. J. Kuo. Shape from shading with a generalized reflectance map model. *Computer Vision and Image Understanding*, 67(2):143–160, 1997.

[38] M. F. Tappen, W. T. Freeman, and E. H. Adelson. Recovering intrinsic images from a single image. *Pattern Analysis and Machine Intelligence*, 27(9):1459–1472, 2005.

[39] P. Fua and Y. G. Leclerc. Object-centered surface reconstruction: combining multi-image stereo and shading. *International Journal of Computer Vision*, 16(1):35–56, 1995.

[40] D. Samaras and D. Metaxas. Incorporating illumination constraints in deformable models. *Computer Vision and Pattern Recognition*, pages 322–329, 1998.

[41] E. Prados, F. Camilli, and O. Faugeras. A unifying and rigorous shape from shading method adapted to realistic data and applications. *Mathematical Imaging and Vision*, 25(3):307–328, 2006.

[42] M. S. Langer and S. W. Zucker. What is a light source? *Computer Vision and Pattern Recognition*, pages 172–178, 1997.

[43] Y. Tian, H. T. Tsui, S. Y. Yeung, and S. Ma. Shape from shading for multiple light sources. *Optical Society of America*, 16(1):36–52, 1999.

[44] K. Barnard and G. Finlayson. Shadow identification using colour ratios. *Color Imaging Conference: Color Science and Engineering Systems, Technologies, Applications*, 8:97–101, 2000.

[45] S. K. Nayar, K. Ikeuchi, and T. Kanade. Shape from interreflections. *International Journal of Computer Vision*, 3:2–11, 1990.

[46] P. N. Belhumeur, D. J. Kriegman, and A. L. Yuille. The bas-relief ambiguity. *International Journal of Computer Vision*, 35(1):33–44, 1999.

[47] D. Scharstein and R. Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *International Journal of Computer Vision*, 47(1-3):7–42, 2002.

[48] D. Scharstein and R. Szeliski. High-accuracy stereo depth maps using structured light. *Computer Vision and Pattern Recognition*, 1:195–202, 2003.

[49] N. D. F. Campbell, G. Vogiatzis, C. Hernandez, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. *European Conference on Computer Vision*, 10(1):766–779, 2008.

[50] S. Baker, R. Szeliski, and P. Anandan. A layered approach to stereo reconstruction. *Computer Vision and Pattern Recognition*, pages 434–441, 2003.

[51] M.Z. Brown, D. Burschka, and G. D. Hager. Advances in computational stereo. *Pattern Analysis and Machine Intelligence*, 25(8):993–1008, 2003.

[52] U. R. Dhond and J. K. Aggarwal. Structure from stereo - a review. *Transactions on Systems, Man, and Cybernetics*, 19(6):1489–1509, 1989.

[53] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002.

[54] Z-F. Wang and Z-G. Zheng. A region based stereo matching algorithm using cooperative optimization. *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[55] A. Klaus, M. Sormann, and K. Karner. Segment-based stereo matching using belief propagation and a self-adapting dissimilarity measure. *International Conference on Pattern Recognition*, 18(3):15–18, 2006.

[56] Q. Yang, L. Wang, R. Yang, H. Stewenius, and D. Nister. Stereo matching with color-weighted correlation, hierachical belief propagation and occlusion handling. *Pattern Analysis and Machine Intelligence*, 31(3):492–504, 2009.

[57] D. Marr and T. Poggio. Cooperative computation of stereo disparity. *M.I.T. A.I Memo*, 364, 1976.

[58] Y. Tsin, S. B. Kang, and R. Szeliski. Stereo matching with reflections and translucency. *Computer Vision and Pattern Recognition*, 1:702–709, 2003.

[59] Y. Taguchi, B. Wilburn, and C. L. Zitnick. Stereo reconstruction with mixed pixels using adaptive over-segmentation. *Computer Vision and Pattern Recognition*, pages 1–8, 2008.

[60] A. L. Yuille and T. Poggio. A generalized ordering constraint for stereo correspondence. *M.I.T. A.I Memo*, 777, 1984.

[61] J. Sun, Y. Li, S.B. Kang, and H. Y. Shum. Symmetric stereo matching for occlusion handling. *Computer Vision and Pattern Recognition*, 2:399–406, 2005.

[62] S. Lin, Y Li, S. B. Kang, X. Tong, and H. Y. Shum. Diffuse-specular seperation and depth recovery from image sequences. *Lecture Notes in Computer Science*, 2352:210–224, 2002.

[63] M. Bleyer and M. Gelautz. A layered stereo matching algorithm using image segmentation and global visibility constraints. *Image Processing*, 5:2997–3000, 2004.

[64] A. Baumberg. Reliable feature matching across widely separated views. *Computer Vision and Pattern Recognition*, 1:774–781, 2000.

[65] R. Zabih and J. Woodfill. Non-parametric local transforms for computing visual correspondence. *European Conference on Computer Vision*, 2:151–158, 1994.

[66] M. H. Lin. *Surfaces With Occlusions From Layered Stereo*. PhD thesis, Stanford University, 2002.

[67] Y. Yang, A. Yuille, and J. Lu. Local, global, and multilevel stereo matching. *Computer Vision and Pattern Recognition*, pages 274–279, 1993.

[68] A. F. Bobick and S. S. Intille. Large occlusion stereo. *International Journal of Computer Vision*, 33(3):181–200, 1999.

[69] S. Birchfield and C. Tomasi. A pixel dissimilarity measure that is insensitive to image sampling. *Pattern Analysis and Machine Intelligence*, 20(4):401–406, 1998.

[70] S. Birchfield and C. Tomasi. Depth discontinuities by pixel-to-pixel stereo. *International Journal of Computer Vision*, 35(3):269–293, 1999.

[71] R. Szeliski and D. Scharstein. Symmetric sub-pixel stereo matching. *Symmetric Sub-Pixel Stereo Matching*, 7(2):525–540, 2002.

[72] B. K. P. Horn and B. G. Schunck. Determining optical flow. *MIT AI Memo*, 572, 1980.

[73] C. Harris and M. J. Stephens. A combined corner and edge detector. *Alvey Vision Conference*, pages 147–152, 1988.

[74] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981.

[75] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *Pattern Analysis and Machine Intelligence*, 12(9):855–867, 1990.

[76] Y. Boykov, O. Veksler, and R. Zabih. Fast approximate energy minimization via graph cuts. *Pattern Analysis and Machine Intelligence*, 23(11):1222–1239, 2001.

[77] J. S. Yedidia, W. T. Freeman, and Y. Weiss. *Understanding Belief Propagation and its Generalizations*. Morgan Kaufmann, 2003.

[78] I. J. Cox, S. L. Hingorani, S. B. Rao, and B. M. Maggs. A maximum likelihood stereo algorithm. *Computer Vision and Image Understanding*, 63(3):542–567, 1996.

[79] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient belief propagation for early vision. *Computer Vision and Pattern Recognition*, 1:261–268, 2004.

[80] M. F. Tappen and W. T. Freeman. Comparison of graph cuts with belief propagation for stereo, using identical mrf parameters. *International Conference on Computer Vision*, 9(2):900–906, 2003.

[81] M. Isard. Pampas: Real-valued graphical models for computer vision. *Computer Vision and Pattern Recognition*, (1):613–620, 2003.

[82] L. Xu and J. Jia. Stereo matching: An outlier confidence approach. *European Conference on Computer Vision*, 10.

[83] Q. Yang, R. Yang, J. Davis, and D. Nister. Spatial-depth super resolution for range images. *Computer Vision and Pattern Recognition*, pages 1–8, 2007.

[84] M. Okutomi and T. Kanade. A multiple-baseline stereo. *Pattern Analysis and Machine Intelligence*, 15(4):353–363, 1993.

[85] K. N. Kutulakos and S. M. Seitz. A theory of shape by space carving. *International Journal of Computer Vision*, 38(3):199–218, 2000.

[86] S.M. Seitz and C. R. Dyer. Photorealistic scene reconstruction by voxel coloring. *Computer Vision and Pattern Recognition*, pages 1067–1073, 1997.

[87] L. Wang, M. Liao, M. Gong, R. Yang, and D. Nister. High-quality real-time stereo using adaptive cost aggregation and dynamic programmming. *Symposium on 3D Data Processing, Visualization, and Transmission*, 3:798–805, 2006.

[88] A. Blake, A. Zisserman, and G. Knowles. Surface descriptions from stereo and shading. *Image Vision Computation*, 3(4):183–191, 1985.

[89] M. G. H. Mostafa, S. M. Yamany, and A. A. Farag. Integrating stereo and shape from shading. *Image Processing*, 3:130–134, 1999.

[90] D. R. Hougen and N. Ahuja. Adaptive polynomial modelling of the reflectance map for shape estimation from stereo and shading. *Computer Vision and Pattern Recognition*, pages 991–994, 1994.

[91] D. R. Hougen. *Estimation Of Shape, Lighting And Reflectivity From Stereo And Shading Information*. PhD thesis, University of Illinois, 1994.

[92] D. Samaras, D. Metaxas, P. Fua, and Y. G. Leclerc. Variable albedo surface reconstruction from stereo and shape from shading. *Computer Vision and Pattern Recognition*, 1:480–487, 2000.

[93] D. Samaras and D. Metaxas. Incorporating illumination constraints in deformable models for shape from shading and light direction estimation. *Pattern Analysis and Machine Intelligence*, 25(2):247–264, 2003.

[94] H. Fassold, R. Danzl, K. Schindler, and H. Bischof. Reconstruction of archaeological finds using shape from stereo and shape from shading. *Procedings 9th Computer Vision Winter Workshop*, 2004.

[95] M. Shao, R. Chellappa, and T. Simchony. Note reconstructing a 3-d depth map from one or more images. *Computer Vision, Graphics, and Image Processing: Image Understanding*, 53(2):219–226, 1991.

[96] H. Jin, A. Yezzi, and S. Soatto. Stereoscopic shading: Integrating multiframe shape cues in a variational framework. *Computer Vision and Pattern Recognition*, 1:169–176, 2000.

[97] A. P. Pentland. Finding the illuminant direction. *Optical Society of America*, 72(4):448–455, 1982.

[98] W. Chojnacki, M. J. Brooks, and D. Gibbins. Revisiting pentland's estimator of light source direction. *Optical Society of America*, 11(1):118–124, 1994.

[99] P. Nillius and J-O. Eklundh. Automatic estimation of the projected light source direction. *Computer Vision and Pattern Recognition*, 1:1076–1083, 2001.

[100] C-Y. Kim, A. P. Petrov, H-K. Choh, Y-S. Seo, and I-S Kweon. Illuminant direction and shape of a bump. *Optical Society of America, A*, 15(9):2341–2350, 1998.

[101] D. Samaras and D. Metaxas. Coupled lighting direction and shape estimation from single images. *International Conference on Computer Vision*, 2:868–874, 1999.

[102] K. Ikeuchi and K. Sato. Determining reflectance properties of an object using range and brightness images. *Pattern Analysis and Machine Intelligence*, 13(11):1139–1153, 1991.

[103] Y. Zhang and Y-H. Yang. Illuminant direction determination for multiple light sources. *Computer Vision and Pattern Recognition*, 1:269–276, 2000.

[104] Y. Zhang and Y-H. Yang. Multiple illuminant direction detection with application to image synthesis. *Pattern Analysis and Machine Intelligence*, 23(8):915–920, 2001.

[105] Y. Wang and D. Samaras. Estimation of multiple illuminants from a single image of arbitary known geometry. *European Conference on Computer Vision*, pages 272–288, 2002.

[106] C-S. Bouganis and M. Brookes. Multiple light source detection. *Pattern Analysis and Machine Intelligence*, 26(4):509–514, 2004.

[107] A. Ortiz and G. Oliver. Estimation of directional and ambient illumination parameters by means of a calibration object. *Lecture Note in Computer Science*, 3211:672–679, 2004.

[108] M. W. Powell, S. Sarkar, and D. Goldgof. A simple strategy for calibrating the geometry of light sources. *Pattern Analysis and Machine Intelligence*, 23(9):1022–1027, 2001.

[109] W. Zhou and C. Kambhamettu. Estimation of illuminant direction and intensity of multiple light sources. *European Conference on Computer Vision*, pages 206–220, 2002.

[110] W. Zhou and C. Kambhamettu. Estimation of the size and location of multiple area light sources. *International Conference on Computer Vision*, 3:214–217, 2004.

[111] W. Zhou and C. Kambhamettu. A unified framework for scene illuminant estimation. *Image and Vision Computing*, 26(3):415–429, 2008.

[112] M. Weber and R. Cipolla. A practical method for estimation of point light-sources. *British Machine Vision Conference*, 2:471–480, 2001.

[113] P. Lagger and P. Fua. Retrieving multiple light sources in the presence of specular reflections and texture. *Computer Vision and Image Understanding*, 111(2):207–218, 2008.

[114] K. Hara, K. Nishino, and K. Ikeuchi. Light source position and reflectance estimation from a single view without the distant illumination assumption. *Pattern Analysis and Machine Intelligence*, 27(4):493–505, 2005.

[115] K. Hara, K. Nishino, and K. Ikeuchi. Multiple light sources and reflectance property estimation based on a mixture of spherical distributions. *International Conference on Computer Vision*, 2:1627–1634, 2005.

[116] K. Hara, K. Nishino, and K. Ikeuchi. Mixture of spherical distributions for single-view relighting. *Pattern Analysis and Machine Intelligence*, 30:25–35, 2008.

[117] I. Sato, Y. Sato, and K. Ikeuchi. Illumination distribution from shadows. *Computer Vision and Pattern Recognition*, pages 306–312, 1999.

[118] I. Sato, Y. Sato, and K. Ikeuchi. Illumination distribution from brightness in shadows: Adaptive estimation of illumination distribution with unknown reflectance properties in shadow regions. *International Conference on Computer Vision*, 2:875–882, 1999.

[119] I. Sato, Y. Sato, and K. Ikeuchi. Stability issues in recovering illumination distribution from brightness in shadows. *Computer Vision and Pattern Recognition*, 2:400–407, 2001.

[120] Y. Wang and D. Samaras. Estimation of multiple directional light sources for synthesis of mixed reality images. *Pacific Conference on Computer Graphics and Applications*, pages 38–47, 2002.

[121] Y. Wang and D. Samaras. Multiple directional illuminant estimation from a single image. *Color and Photometric Method in Computer Vision Workshop*, 2003.

[122] Y. Wang and D. Samaras. Estimation of multiple directional illuminants from a single image. *Image and Vision Computing*, 26:1179–1195, 2008.

[123] Y. Li, S. Lin, H. Lu, and H-Y. Shum. Multiple-cue illumination estimation in textured scenes. *International Conference on Computer Vision*, 2:1366–1373, 2003.

[124] T. Okabe, I. Sato, and Y. Sato. Spherical harmonics vs. haar wavelets: Basis for recovering illumination from cast shadows. *Computer Vision and Pattern Recognition*, 1:50–57, 2004.

[125] B. V. Funt, M. S. Drew, and M. Brockington. Recovering shading from color images. *European Conference on Computer Vision*, 2:124–132, 1992.

[126] Y. Weiss. Deriving intrinsic images from image sequences. *International Conference on Computer Vision*, 2:68–75, 2001.

[127] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Constructing free-energy approximations and generalized belief propagation algorithms. *Information Theory*, 51(7):2282–2312, 2005.

[128] Y. Weiss and W. T. Freeman. Correctness of belief propagation in gaussian graphical models of arbitrary topology. *Neural Computation*, 13(10):2173–2200, 2001.

[129] R. Cowell. *Learning in Graphical Models*, chapter Advanced Inference in Bayesian Networks. MIT Press, 1998.

[130] D. Bickson, D. Dolev, O. Shental, P. H. Siegel, and J. K. Wolf. Linear detection via belief propagation. *Allerton Conference on Communications, Control and Computing*, 45, 2007.

[131] O. Shental, D. Bickson, P. H. Siegel, J. K. Wolf, and D. Dolev. Gaussian belief propagation solver for systems of linear equations. *International Symposium on Information Theory*, pages 1863–1867, 2008.

[132] D. Nehab, S. Rusinkiewicz, J. Davis, and R. Ramamoorthi. Efficiently combining positions and normals for precise 3d geometry. *SIGGRAPH*, 24(3):536–543, 2005.

[133] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

[134] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. *SIGGRAPH*, pages 369–378, 1997.

[135] K. V. Mardia and P. E. Jupp. *Directional Statistics*. Wiley, 2000.

[136] R. A. Maronna, R. D. Martin, and V. J. Yohai. *Robust Statistics*. Wiley, 2006.

[137] N. I. Fisher. Robust estimation of the concentration parameter of fisher's distribution on the sphere. *Applied Statistics*, 31(2):152–154, 1982.

[138] A. H. Land and A. G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28(3):497–520, 1960.

[139] R. Hartley and A. Zisserman. *Multiple View Geometry in computer vision*. Cambridge University Press, 2nd edition, 2003.

[140] Z. Zhang. A flexible new technique for camera calibration. *Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[141] C. Loop and Z. Zhang. Computing rectifying homographies for stereo vision. *Computer Vision and Pattern Recognition*, 1:125–131, 1999.

[142] M. Pollefeys, R. Koch, and L. V. Gool. A simple and efficient rectification method for general motion. *International Conference on Computer Vision*, 1:496–501, 1999.

[143] F. E. Nicodemus. Directional reflectance and emissivity of an opaque surface. *Applied Optics*, 4(7):767–773, 1965.

[144] P. Dutre, P. Bekaert, and K. Bala. *Advanced Global Illumination*. A K Peters, 2003.

[145] K. E. Torrance and E. M. Sparrow. Theory for off-specular reflection from roughened surfaces. *Optical Society of America*, 57(9):1105–1114, 1967.

[146] G. J. Ward. Measuring and modeling anisotropic reflection. *SIGGRAPH*, 26(2):265–272, 1992.

[147] E. Sudderth. *Graphical Models for Visual Object Recognition and Tracking*. PhD thesis, Massachusetts Institute of Technology, 2006.

[148] D. Sharon. Loopy belief propagation in image-based rendering. Technical report, University of British Columbia.

[149] E. B. Sudderth, A. T. Ihler, W. T. Freeman, and A. S. Willsky. Nonparametric belief propagation. *Computer Vision and Pattern Recognition*, 1:605–612, 2003.

[150] A. L. Yuille. Cccp algorithms to minimize the bethe and kikuchi free energies: Convergant alternatives to belief propagation. *Neural computation*, 14(7):1691–1722, 2006.

[151] Y. Weiss and W. T. Freeman. On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs. *Information Theory*, 47(2):736–744, 2001.

[152] A. T. Ihler, J. W. Fisher III, and A. S. Willsky. Message errors in belief propagation. *Advances in Neural Information Processing Systems*, 2005.

[153] D. J. C. MacKay. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 2003.

[154] J. S. Yedidia, W. T. Freeman, and Y. Weiss. Generalized belief propagation. *Neural Information Processing Systems*, 13:689–695, 2001.

[155] T. Heskes, K. Albers, and B. Kappen. Approximate inference and constrained optimisation. *Uncertainty in Artificial Intelligence*, pages 313–320, 2003.

[156] S. Calderara, R. Cucchiara, and A. Prati. Detection of abnormal behaviors using a mixture of von mises distributions. *Advanced Video and Signal Based Surveillance*, pages 141–146, 2007.

[157] A. Banerjee, I. S. Dhillon, J. Ghosh, and S. Sra. Clustering on the unit hypersphere using von mises-fisher distributions. *Machine Learning Research*, 6:1–39, 2005.

[158] J. T. Kent. The Fisher-Bingham distribution on the sphere. *Royal Statistical Society, Series B*, 44(1):71–80, 1982.

[159] R. Fisher. Dispersion on a sphere. *Proceedings of the Royal Society of London, Series A, Mathematical and Physical Sciences*, 217:295–305, 1953.

[160] C. Bingham and K. V. Mardia. A small circle distribution on the sphere. *Biometrika*, 65(2):379–389, 1978.

[161] A. Kume and A. T. A. Wood. Saddlepoint approximations for the Bingham and Fisher-Bingham normalising constants. *Biometrika*, 92(2):465–476, 2005.

[162] J. C. Hart. Distance to an ellipsoid. *Graphics Gems IV*, pages 113–119, 1994.